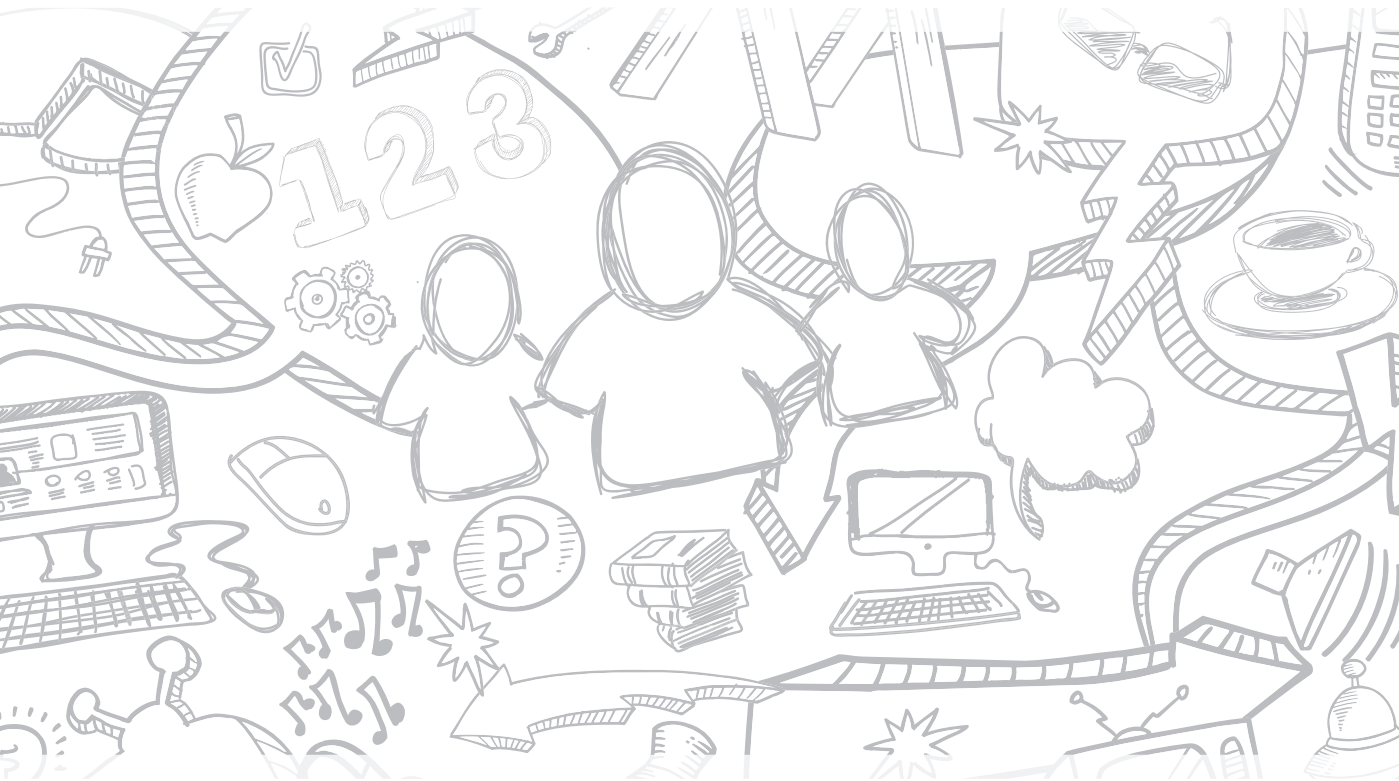
安全技术大系

Kali Linux 大揭秘

深入掌握渗透测试平台



Kali Linux Revealed

Mastering the Penetration Testing Distribution

[美] Raphael Hertzog Jim O'Gorman Mati Aharoni 著

诸葛建伟 王珩 刘跃 梁智溢 译

電子工業出版社

Publishing House of Electronics Industry

北京•BEIJING

内 容 简 介

Kali Linux 是设计用于数字取证和渗透测试的操作系统。本书是官方出版的唯一著作，适合新手入门。

在本书中，重点介绍了 Kali Linux 平台本身，并详细论述了如何来理解和最大限度地使用 Kali。本书首先引导读者了解 Kali Linux 的功能和基础知识，并解释了基本的 Linux 命令和概念。然后介绍了最常见的 Kali Linux 安装方案，并探讨了如何配置、分析和保护 Kali Linux。随后介绍了强大的 Debian 软件包管理器，在这部分中，介绍了如何安装和配置软件包，如何更新和升级 Kali 安装，以及如何创建自定义软件包，并介绍了如何在大型企业网络中部署自定义安装。最后，进入到高级主题，介绍了内核编译、自定义 ISO 创建、工业强度加密，以及如何安装加密杀死开关来保护敏感信息等内容。

无论你是老将还是新手，本书都是你学习 Kali Linux 的最佳选择。

版权贸易合同登记号　图字：01-2018-0971

图书在版编目（CIP）数据

Kali Linux 大揭秘：深入掌握渗透测试平台 /（美）拉斐尔·赫佐格（Raphael Hertzog），（美）吉姆·奥戈曼（Jim O'Gorman），（美）马蒂·艾哈尼（Mati Aharoni）著；诸葛建伟等译. —北京：电子工业出版社，2018.8
（安全技术大系）
书名原文：Kali Linux Revealed: Mastering the Penetration Testing Distribution
ISBN 978-7-121-34313-1

Ⅰ. ①K… Ⅱ. ①拉… ②吉… ③马… ④诸… Ⅲ. ①Linux 操作系统－安全技术 Ⅳ. ①TP316.85

中国版本图书馆 CIP 数据核字(2018)第 115548 号

责任编辑：刘　皎
印　　刷：
装　　订：
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编：100036
开　　本：787×980　1/16　印张：18.5　字数：390 千字
版　　次：2018 年 8 月第 1 版
印　　次：2018 年 8 月第 1 次印刷
定　　价：89.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：010-51260888-819，faq@phei.com.cn。

推荐序一

应诸葛建伟先生之邀，为其翻译的《Kali Linux 大揭秘：深入掌握渗透测试平台》一书作序。OWASP 作为国际权威应用安全的研究机构，在 Web 安全方面的研究，是国内外信息安全机构在应用安全研究方面的主要参考依据。本人作为 OWASP 中国北京的主要负责人，很荣幸地拜读了本书的整个目录和全部章节，最大的感受就是本书贴合实际需求、生动详实，案例充分。

本书从渗透测试的实战出发，增加了诸多新增工具的介绍，完整地填补了目前市面上相关书籍内容上的空白：现在市面上的许多安全书籍，都是只介绍结果，考虑过程的并不多。本书从实践出发，本着务实的精神，对环境搭建一直到渗透测试的全阶段，以及主流工具的使用，都做了详尽的介绍，示例丰富，是每位信息安全的从业人员、在校学生不可多得的一本使用大全。你完全可以依照本书的案例来学习，并有效地贴近企业需求，做到有的放矢。

从本书的编写风格就可以看出作者技术功底扎实、写作思路清晰、讲解由浅入深、举例生动详实，非常值得一读！

陈　亮
OWASP 中国

推荐序二

你可能还没意识到你所拥有的美好事物。

在 1998 年，我还是一位初出茅庐的黑客，创建了最早的一个职业白帽黑客团队。当时团队成员们还都是一群孩子，梦想着自己未来的工作，能够拿着不错的薪水，去入侵这个星球上最安全的计算机系统、网络甚至建筑。

这听起来很“性感”，但现实却是，我们需要把生活中的绝大部分时间花在和电脑键盘相处上，用行业中的一些数字工具来武装自己。我们挥舞着四处采集到的程序，绘制网络地图并定位目标，然后扫描、渗透目标并进一步拓展。在某些情况下，我们中的一员（经常是 Jim Chapple）会编写一些定制化工具来做一些很酷的事情，比如说扫描一个 A 类的网络（在当时可没有其他现成工具能够搞定这件事），但绝大多数时候我们只是使用和修改黑客社区中已开发好的工具。在那些还没有 Google 的日子里，我们频繁地刷 BugTraq、AstaLaVista、Packet Storm、w00w00、SecurityFocus、X-Force 以及其他资源，来研究并建立起我们自己的军火库。

因为我们在每次行动中的时间有限，我们必须追求速度，这意味着我们不会花很多时间来拨弄各种工具。所以我们必须要学习最核心工具的里里外外，然后让其他辅助工具只是处于待命状态。于是我们必须良好地组织我们的工具，文档化并进行测试，以便在实战中不要给我们带来意外。如果不能成功渗透，那么我们将在客户面前丢脸，而他们也会轻视我们所提出的建议。

正因为如此，我花了非常多的时间来对工具进行分类编目，当一个工具被发布或更新时，我会完整地走完一个流程，查看它是否可以在攻击平台上正常运行，以及是否值得安装运行。我还必须更新依赖这一工具的所有脚本，编写相关文档并进行测试，包括相对于前一版本有了何种变化。然后我会组织所有的工具，根据它们在执行一次安全评估时的目的和作用，将它们放置到不同的目录里。我会为特定工具编写包装脚本，将一些工具链接在一起，并将所有工具关联在一起，刻录成一张 CD 盘，当客户不让我们带入攻击机器或者移动介质到他们实验室的时候，我们将带着这个 CD 盘进入隔离的敏感区域。这一过程是非常让人头疼的，但也是必须的。我们知道，如果合适地应用我们的技能和专业经验，保持良好的组织

性，并高效工作，我们有入侵任何网络的能力。尽管在渗透测试中从不失手是我们的驱动力，但是我们提供给客户的是一种专业入侵网络的渗透服务，而客户会增加各种限制，并通过金钱回报引导渗透的主要目标到他们最为关键但可能疏于防范的信息安全流程中。

我们花了好几年的时间来磨练我们的技能和专业度，但如果没有良好的组织和高效率，我们不会像现在这样成功。如果我们没有尽心去调校所需的工具，我们可能已经失败了。这是我花那么多时间研究、编撰文档、测试与分类工具的根本原因。在 21 世纪到来之后，这很快发展成为一份压力巨大的全职工作。由于 Internet 的发展，攻击面的全球性爆发，以及攻击工具的种类和数量的指数级增长，维护这些工具所需的工作量也随之剧增。

从 2004 年开始，Internet 已经不仅仅是商务活动的基石，也成为了一个社交平台。计算机已经变成人人都能负担得起的东西，而且更加友好，无所不在。存储技术已经从 M 字节扩展到 G 字节，以太网的速率从每秒几百 KB 快速发展到每秒数十 MB，Internet 连接变得越来越快，也更加便宜。电子商务已经非常流行，而社交媒体如 Facebook（2004 年）、Twitter（2006 年）已经上线，Google 变得更加成熟，可以让每个人（包括罪犯）在互联网上找到几乎任何东西。

研究对于我们这样的团队变得至关重要，因为我们必须能够跟上新型攻击和工具集发展的脚步。我们对更多计算机犯罪事件进行响应，在必要的取证工作中我们需要仔细追查潜在的证据。而自生型 CD 的概念意味着我们可以在被攻陷的机器上执行实时取证，而不会对证据造成任何破坏。现在我们这个小团队必须要管理攻击工具、取证工具和在敏感区域使用工具发行版，我们必须跟上所有最新攻击技术和利用方法学的发展。我们必须这么做，因为你知道我们是受雇来进行渗透测试的，所以都是高标准要求。而事情变得有些超出控制，在最近一段时间，我们在实战中花费的时间少了，而在研究、打磨工具和计划方面投入更多的时间。

在这场对抗中我们并不孤独，在 2004 年，Mati“Muts”Aharoni，一位黑客和安全专家，发布了“WHoppiX” (White Hat Knoppix)，一个被宣传为“终极版渗透测试 Live CD”的自生 Linux CD，其中包含了“来自 SecurityFocus、Packet Storm 和 kotik 的所有渗透利用脚本，Metasploit 框架 2.2 版，以及更多其他软件”。记得在下载 WHoppiX 时，我在想这是很棒的东西，我也下载了其他一些自生 CD，想象假使我之前陷入困境的一些局面，这些自生 CD 可以在实战中帮上我们。但是我当时并没有计划在真正实战工作中依赖 WHoppiX 和其他 CD。我并不相信它们中的任何一个能够满足我的大部分需求，没有一个对我的工作流来说感觉是对的，而且它们也不是完全的、可安装的发行版。在我下载的时候，它们已经过时了。一个过时的工具集对于我们所在的行业就是“死神降临”。我只是简单地将这些 CD 镜像添加到军火库里，尽管它们的大小相对较大，然后还是继续着那令人头疼的过程来维护我们实际使用的工具集。

尽管在当时我对它持保留意见，但 WHoppiX 及其后续者在我们的行业和社区中都造成了巨大的影响，或许也超出了 Muts 本人的预期。

在 2005 年，WHoppiX 进化成了 WHAX，一个扩展并更新后的工具集，基于“最模块化的 SLAX（Slackware）自生 CD”。Muts 和来自黑客社区的一个迅速发展的志愿者团队看起来已经意识到了，无论他们如何具有远见，仍然无法预期到我们这个行业的所有发展和波动，而 CD 的用户们在实战中也有多种差异化需求。而且 Muts 和他的团队显然也已经在实战中使用 WHAX，他们似乎也努力让它变得更加好用，这件事对于我是非常受鼓舞的。

到了 2006 年，Muts、Max Moser 和他们的团队将 Auditor Security Linux 和 WHAX 合并到单一发行版中，称为 BackTrack。基于 SLAX，BackTrack 继续向前发展，增加了更多的工具，更多的框架，扩展语言支持，多种无线协议支持，整合一个对专业用户和新手用户都适用的菜单结构，以及重度修改了内核。BackTrack 成为领先的安全发行版，但是像我这样的很多人仍然把它作为“实际使用工具集”的一个备份。

在 2009 年初，Muts 和他的团队已经将 BackTrack 扩展至 BackTrack 4，这是一个明显的变化。今天，Muts 将此作为了全职工作，BackTrack 已经不再是一个自生 CD，而是一个基于 Ubuntu 充分发展的发行版，并充分利用了 Ubuntu 的软件仓库。这种变化以一个显著的革新作为标志：也就是 BackTrack 4 拥有系统更新机制。用 Muts 自己的话说：“当和我们的 BackTrack 的软件仓库同步时，你可以在安全工具发布后很快就获得更新。”这真的是一个转折点，BackTrack 团队已经真正理解了渗透测试者、取证分析师和该行业其他从业者所面对的挑战，他们的工作可以帮助我们节省很多时间，并为我们提供了一个坚实的基础，使得我们直接进入实战状态，将更多的时间花在真正重要且有趣的事情上。结果是，社区反响热烈，众多用户群集到论坛和 wiki 上，并热情参与到开发团队中。BackTrack 确实是一个社区化的贡献，而 Muts 仍在其中引领着。

BackTrack 4 最终成为工业界最强的渗透测试平台，而我，还有很多像我这样的人终于可以松一口气了。我们对 Muts 和他团队所承受的“痛苦”有着切身体会，因为我们也曾经亲身经历过。而结果是，我们中的很多人开始使用 BackTrack 作为工作的首要基础平台。是的，我们仍然会拨弄一些工具，编写我们自己的代码，以及开发我们自己的漏洞利用工具和技术，我们开展研究和实验，但是我们不再在收集、更新和组织工具上花费很多时间了。

BackTrack 4 R1 和 R2 是 2010 年进一步修订的版本，之后又在 2011 年自底向上重新构建了 Back Track 5——仍然基于 Ubuntu，但跟上每个发布的脚步。Back Track 在当时已经是一个极为成功的项目，需要英雄的志愿者团队和社区的努力，同时也需要一些资助。Muts 在 2006 年开始创立 Offensive Security 公司，该公司不仅仅提供全球领先的培训和渗透测试服务，还为保持 Back Track 的开发进程提供坚实后盾，保证 Back Track 的开源和免费使用。

Back Track 在 2012 年继续发展和改进（先后发布 R1、R2 和 R3 版本），其维护着 Ubuntu 的核心并增加了数以百计的新工具，包括物理和硬件利用工具、VMware 支持、许多无线和硬件驱动，以及大量的稳定性改进和缺陷修复。然而，在 R3 版本发布之后，Back Track 的开发变得相对缓慢，甚至有点神秘地静默。

当时在坊间有一些猜测，一些人以为 Back Track 变了，把它的灵魂出售给了一家无情邪恶的企业主而获得丰厚回报。Offensive Security 也在那几年里发展成为最受尊敬的网络安全培训机构之一以及行业中的一家思想引领者，某些人揣测这家公司的成功是建立在吞并大量 Back Track 关键开发者的基础上。然而，事实并非如此。

2013 年，Kali Linux 1.0 发布，从发布公告中可以看到，经过一年沉默的开发过程，Offensive Security 自豪地宣布，发布 Kali Linux，并且它是公开可用的，这是最先进、鲁棒和稳定的渗透测试发行版，Kali 是比 Back Track 更加成熟、安全和适合企业应用的版本。

Kali Linux 于 Back Track 并不是“新瓶装旧酒”，其移植了超过 600 个完全重打包后的工具，这显然是一个令人称奇的工具集，但还不仅仅只是这些。Kali 是从 Debian 的内核核心基础上，从头开始构建的，这对于一些还不了解情况的人而言，看起来并不是个大工程，但是懂行的人知道这需要做大量底层的工作。做了这些重打包的工作后，Kali 用户可以下载每一个工具的源码包，可以按需求修改并重构工具，而只需要敲几行代码和命令。和目前其他主流操作系统发行版不同的是，Kali Linux 同 Debian 软件仓库每天进行四次同步，这意味着 Kali 用户可以及时获取软件包更新和安全补丁。Kali 开发者全身心投入到许多工具上游版本的打磨、打包和维护上，才使得用户可以始终保持在业界前沿。感谢 Debian 的根基，Kali 用户可以从软件仓库直接引导一个安装或者 ISO 镜像，这也打开了完全定制化 Kali 安装版之门，你可以通过预设文件进行自动化和定制的大规模的企业化部署。为了实现定制目标，Kali 用户可以修改桌面环境，变更菜单项，改变图标文件，甚至改变窗口环境。由于做了大量的 ARM 开发工作，使得 Kali Linux 可以在大范围硬件平台上安装，包括无线 AP、单片机系统（例如树莓派、ODROID、BeagleBone、CubieBoard 等），以及基于 ARM 的 Chromebook 计算机等。最后要提及的重要一点是，Kali Linux 支持无缝的小更新和重大升级，这意味着 Kali 的狂热者永远不需要在机器上重新安装定制化的 Kali Linux。

社区也注意到了 Kali 的发布，在最初的五天中，9 万名用户下载了 Kali 1.0 版本，而这仅仅是开端。在 2015 年，Kali 2.0 发布，紧随其后的是 2016 年的 rolling 版本发布。总结来说，如果 Kali 1.0 聚焦于构建一个稳固的基础设施，那么 Kali 2.0 则专注于提升用户体验并维护更新的软件包和工具仓库。而目前 Kali Linux 的版本是一种滚动更新的发行版，这标志着特定版本的终结。

现在用户可以持续地保持最新版本，并接收到开发者创建的更新和补丁。得益于上游软

件的版本标记系统，核心工具的更新更加频繁，也针对视障人士做了便捷性方面的改进，Linux 内核也被进行了更新和修补以扩展对无线 802.11 注入的支持范围。SDR（Software Defined Radio，软件定义无线电）和 NFC（Near-Field Communication，近场通信）工具增加了对安全测试新领域的支持，完全 Linux 加密磁盘安装和紧急状况自毁选项也变得可用。感谢 LVM、LUKS，增加了 USB 持久化选项，并允许基于 USB 的 Kali 安装保持重启后的系统改变，而无论 USB 盘是否加密。最后，最新版本的 Kali 为 NetHunter 打开了大门，NetHunter 成为一个运行在移动设备上的全球领先的开源操作系统，以 Kali Linux 和 Android 作为其基础。

Kali Linux 已经演化成为信息安全从业者首选的平台，而且真正成为了一个工业级别、全球领先、成熟、安全、适合企业使用的操作系统发行版。通过十多年的开发历程，Muts 和他的团队，以及来自黑客社区无法计数的全身心投入的志愿者，承担了组织我们工作环境的压力，把我们从大量工作、负担中释放出来，并提供了一个安全和可靠的基础平台，让我们聚焦在驱动工业界向前发展，让数字世界更安全的终极目标上。

围绕着 Kali Linux，一个令人惊奇的社区已经创建起来，每个月新版本的 Kali 被三四十万的用户下载，我们汇聚在 Kali 论坛和 Kali 的 IRC 频道中，我们汇集在 Kali Dojos 会议上，从开发者那里学习如何最好地利用 Kali。

Kali Linux 改变了信息安全的世界，让它变得更好，而 Muts 和他的团队帮我们节省了许多时间，让我们能够将更多时间和精力放在一起驱动工业界的向前发展上。

尽管 Kali 拥有惊人的接受度、支持度和流行度，它却从来没有发布一份官方的手册。好吧，现在这种状况已经改变了，我非常激动地和 Kali 开发团队，特别是 Mati Aharoni、Raphaël Hertzog、Devon Kearns 和 Jim O'Gorman，一起来提供这本指南，第一本关于 Kali Linux 的官方出版物，或许也是官方手册系列中的第一本。在这本书中，我们聚焦于 Kali Linux 平台本身，帮助你从最基础了解和使用 Kali。我们还不会深入到 Kali Linux 中的工具军火库中，但无论你是一位老兵还是一位新手，如果你准备深入探索 Kali Linux，本书是你最好的起点。无论你在这个游戏中已经玩了多长时间，阅读这本书，你将可以和正在发展的 Kali Linux 社区联结在一起，我们行业中最古老、最大、最有生气以及最活泼的社区之一。

我代表 Muts 和 Kali Linux 开发者团队的其他同仁，恭喜你踏上深入掌握 Kali Linux 的征程。

Johnny Long
2017 年 2 月

作 者 序

你所在的渗透测试团队定购的 16 台高性能笔记本电脑刚刚已经到货，而你被赋予设置好工作环境的任务，那就为明天一场驻场渗透测试做好准备吧。你安装好 Kali 系统并启动了其中一台电脑，以确认它是否可用。尽管 Kali 拥有最前沿的内核，但是网卡和鼠标仍然不工作，笔记本电脑中最新款的 NVIDIA 显卡和 GPU 也正在困惑地“注视”着你，因为缺少恰当安装的驱动程序。你很无奈。

在 Kali 的自生模式下，你快速地在一个终端中输入 lspci 命令，然后眯着眼睛盯着屏幕，当你滚动查看硬件列表：“PCI bridge、USB controller、SATA controller、Ethernet 和 Network controllers”，并通过 Google 快速搜索这些硬件相应的型号，以及 Kali 内核版本号的交叉索引时，发现这些最新款硬件的驱动还没有被集成到主线的内核代码中。

这对于你来说并不是难题。一个解决计划正在你的脑袋中成型，而这要归功于你几周前仔细研读过的《Kali Linux 大揭秘》这本书。你使用 Kali Live-Build 系统来创建一个定制 Kali ISO 镜像，将所需的驱动程序灌入安装盘里。另外，除了包含 NVIDIA 显卡驱动之外，还安装了用来发挥 GPU 性能并用于 hashcat 程序的 CUDA 库文件，它使得 GPU 可以在飞速破解哈希口令时发挥最大的功效。你甚至扔了一张带有微软 Logo 的定制桌面进去，来冒充正在使用 Windows 桌面工作的机器。

因为你这次安装的所有机器硬件配置是一致的，所以你可以在 ISO 镜像中增加一个预设的启动选项，这使得你可以从一个 U 盘启动并在没有任何用户干预的情况下安装好 Kali。让安装任务自动完成，并拥有一个全盘加密的系统。

棒极了！你现在已经按照你的需求，为你的机器硬件配置特别设计及优化更新了 Kali 系统版本，你节省了一整天的时间，任务完成！

我们其中的一些人需要从主流操作系统使用者身份中跳出，寻找更加干净、更具意义、更适合于我们工作方式的操作系统，而随着市场上硬件设备的膨胀式发展，上述这样的场景对我们而言变得越来越普遍。

对于投身于网络安全领域的我们更是如此，不管你将网络安全作为一种业余爱好、痴迷的兴趣，还是从事的行业。作为新手，我们经常发现被操作环境或操作系统所困，对于很多

新手而言，Kali 是他们进入 Linux 世界最早接触的系统。我在好几年前就认识到用户群体在这一转变中的困惑，并计划编写一本组织合理的入门引导书来帮助社区用户进入网络安全的世界，而不是一开始直接将所有 Linux 系统的复杂特性全部提供给他们。基于这样的考虑，《Kali Linux 大揭秘》这本书就诞生了。本书让所有通过 Kali Linux 进入网络安全领域的人都能受益。

现在这本书要面世了，然而我们很快意识到还有很多潜藏的“宝藏”有待发掘，因此在这本 Kali Linux 引导书之后，还有很多机会再去探索它更有趣但鲜为人知的特性，就像本书的书名一样，“Kali Linux 大揭秘”。

最后，我们对目前的结果非常开心，本书达到了我们的预期，我可以骄傲地说它已经超出了我们的预期。我们也意识到我们在不经意间扩展了本书的潜在读者群，它不仅仅是为网络安全领域的新手准备的，而且可以帮助一些有经验的渗透测试者提升与改进对 Kali Linux 的控制能力，使得他们可以解密 Kali Linux 发行版中我们准备的各种潜在特性。无论你面对的是一台机器，还是企业中上千台机器；无论是仅仅做配置上的微小修改，还是从内核级别开始完全定制并重建软件仓库；无论是仅仅触及 Debian 软件包管理系统的表层，还是深入探索，《Kali Linux 大揭秘》这本书都提供了路径图。

我代表自己和整个 Kali Linux 团队衷心祝愿，拥有这本导航手册在手，你能够拥有一个激动人心、充满快乐和收获的揭秘之旅！

Muts

2017 年 2 月

前　言

Kali Linux 是世界上最强大和最受欢迎的渗透测试平台，被网络安全领域的专家们广泛使用，其应用领域包括渗透测试、取证、逆向工程及漏洞评估。这是多年来我们精益求精和平台不断发展的结果，从一开始的 WHoppiX 到 WHAX，再到 BackTrack，最终发展到目前融入了 Debian GNU/Linux 以及充满活力的全球开源社区强大能力的完整渗透测试框架平台。

Kali Linux 并不是一个简单的工具集合，而是一个灵活的框架，专业渗透测试人员、安全从业者、学生和业余爱好者都可以根据自己的具体需求，来剪裁和定制它的功能。

为什么要写这本书

Kali Linux 不仅仅是一个在标准 Debian 发行版基础上安装了各种信息安全工具的集合，我们还针对它进行了很多预先的配置工作，以便让你能够立即启动和运行它。为了充分利用 Kali 的功能，你还应当深入了解强大的 Debian GNU/Linux 基础操作，以及学习如何将这些技能使用在你的应用场景中。

Kali 的确有很多用途，但它主要被设计用于渗透测试。本书的目的不仅在于帮助你轻松使用 Kali Linux，更在于提高你对系统的整体认识，使你的操作体验更加流畅，从而在你遇到时间紧迫的渗透测试任务时，无须浪费宝贵时间去安装新软件或者启用新的网络服务。在本书中，我们首先介绍了 Linux，然后更深入一步，介绍了 Kali Linux 的特别之处，以便你准确了解表象之下深层次的特性。

特别是当你时间紧任务重时，这些知识显得非常有价值。掌握这些更深层次的知识有助于你更好地设置 Kali，解决一些疑难问题，按照个人意愿更好地使用工具，分析工具的输出，或者在更大规模的测试场景中使用 Kali。

如果你渴望投身到充满智慧且令人着迷的网络安全领域，并且正好选择了 Kali Linux 作为主要操作平台，那么本书就是为你准备的。本书旨在帮助 Linux 新手以及 Kali 的现有用户加深对 Kali 基础知识的了解，那些已经使用 Kali 多年但希望更加系统深入学习 Kali 的读者，也能够通过本书扩大 Kali 的应用领域，填补之前对 Kali 认知的空白。

此外，本书还可以作为想考取 Kali Linux 认证专家资质读者们的学习大纲、技术参考以及学习指南。

本书的总体思路和结构

这本书的设计非常贴近实战，你从一开始阅读就可以动手去操作 Kali Linux，而不是阅读大量章节的理论介绍。本书对每个主题都以非常务实的方式来介绍，包含了许多样例和图示，以便使讲解更形象和具体。

在第 1 章“关于 Kali Linux”中，我们定义了一些基本术语，并解释了 Kali Linux 的用法。在第 2 章“Kali 入门”中，我们循序渐进地引导你下载 ISO 镜像并在计算机上运行 Kali Linux。第 3 章“Linux 基础”讲述了阅读本书需要了解的 Linux 基本知识，例如 Linux 系统架构、安装过程、文件系统层次结构、权限等重要概念。

至此，你已经使用 Kali Linux 作为自生系统有一段时间了。在第 4 章“安装 Kali Linux”中，将学习如何（在硬盘上）安装永久性的 Kali Linux。在第 5 章“配置 Kali Linux”中将学习如何按照喜好调整它。作为一个已入门的 Kali 用户，现在是熟悉 Kali 中一些重要资源的时候了，第 6 章“自助解决问题并获得帮助”提供了解决可能遇到的各种意外问题的方法。

在掌握了这些基础知识之后，将涉及更高级的主题。第 7 章“加固和监控 Kali Linux”提供了确保 Kali Linux 的安装能够满足你的安全要求的技巧。接下来，第 8 章“Debian 软件包管理”解释了如何充分发挥 Debian 软件包生态系统的潜力。而在第 9 章“高级用法”中，将学习如何创建完全定制的 Kali Linux ISO 镜像。如第 10 章“Kali Linux 企业级应用”中所述，当你在企业规模部署 Kali Linux 时，所有这些主题的内容都可能会用得上。

第 11 章“安全评估简介”使你将在本书中学到的所有知识与安全从业者日常工作联系起来，真正做到学以致用。最后一章，第 12 章展望未来的路。

Raphael Hertzog的致谢

我要感谢 Mati Aharoni！2012 年，当我还是数十名 Debian 顾问中的一员时，他与我联系，说想开发一个基于 Debian 的 BackTrack 替代版本。这就是我从事 Kali Linux 相关工作的开始，从那以后，我就喜欢上了我在 Kali 世界的旅程。

多年来，Kali Linux 越来越向 Debian GNU/Linux 靠近，特别是在基于 Debian 测试版分支的 Kali 滚动发行版之后。现在我的大部分工作，无论是基于 Kali 还是基于 Debian，都在为整个 Debian 生态系统做贡献，这正是我日复一日、年复一年继续下去的动力。

写这本书也是 Mati 给我的一个绝好机会。写书和 Kali 开发是完全不同的工作，但是具有相通的地方：都能够帮助他人并和他们一起分享我们在 Debian/Kali 操作系统领域的专业知识。基于我创作《Debian 管理员手册》的经验，希望我的文字能够帮助你迈出在快速发展的网络安全世界中的第一步。

我还要感谢所有对这本书做出过贡献的 Offensive Security 公司的伙伴们！Jim O'Gorman（一些章节的合著者）、Devon Kearns（评审员）、Ron Henry（技术编辑）、Joe Steinbach 和 Tony Cruse（项目经理）。同时也感谢 Johnny Long 参与撰写序言，并完成了整本书的评审。

Jim O’Gorman的致谢

我要感谢参与这个项目的所有伙伴！我的工作只是其中的一小部分。这本书很像 Kali Linux 本身，是一个由大家共同努力而使工作事半功倍的合作项目。特别感谢 Raphaël、Devon、Mati、Johnny 和 Ron，他们承担了绝大部分工作。没有他们，这本书不会问世。

Mati Aharoni的致谢

Kali Linux 第一个版本从发布到现在已经过去了好几年。从第一天起，我一直梦想着出版一本涵盖整个 Kali 操作系统的官方书籍。因此，我很荣幸能够把这本书奉献给公众。我衷心感谢参与这个项目的所有人，包括 Jim、Devon、Johnny 和 Ron。特别感谢 Raphael 做了本书绝大部分的繁重工作，并将珍贵的专业经验分享给我们的团队。

关于作者

Raphael Hertzog 拥有超过 20 年的 Debian 开发经验，并且是《Debian 管理员手册》的作者，他同时是 Kali 团队中的 Debian 权威。当他还没有进入 Kali 团队工作的时候，他通过自主创业的 Freexian 公司提供 Debian 专家咨询服务，帮助其他公司和个人创建 Debian 的派生定制化安装器和软件包管理系统，改进现有的软件包（bug 修补和增加新特性），等等。

Jim O'Gorman 是 Offensive Security 公司美国安全服务部的总经理，Jim 拥有超过十年在全球范围内对深度防御环境进行渗透测试的经验，此外 Jim 还是 Offensive Security 公司“使用 Kali Linux 进行渗透测试”认证培训的首席讲师。

Mati Aharoni 是信息安全界的一位老兵了，他活跃在安全社区已超过十年。Aharoni 创建了 Back Track 和 Kali 开源发行版，以及 Exploit DB 数据库项目，并创建了 Offensive Security，一家领先的信息安全公司，以工业界领先的安全证书认证和培训闻名于世。在漏洞利用代码开发与编目分类、渗透测试、Kali 开发和捣鼓硬件的过程中，Aharoni 享受着类似于传教士的角色，说服人们聆听关于 Kali Linux 的神奇。

目　录

第1章　关于Kali Linux

内容

- 简要回顾
- 和 Debian 的关系
- 设计目标和使用场景
- Kali Linux 的主要功能
- Kali Linux 的设计策略
- 小结

Kali Linux[1]是基于Debian GNU/Linux的企业级安全审计Linux发行版。Kali的目标用户是安全专业人员和IT管理员，其能够应用于高级渗透测试、取证分析和安全审计等领域。

什么是 Linux 发行版

Linux 虽然经常会被作为整个操作系统的名字，但事实上它特指内核的名字：一个在硬件和最终用户应用之间处理交互的软件。

另一方面，Linux 发行版指的是在 Linux 内核上构建的一整套操作系统，它通常包含一个安装程序和很多应用软件，有些已被预装在系统上，有些被打成软件包，方便随时安装。

Debian GNU/Linux[2]是一个领先的通用Linux发行版，其以质量和稳定性著称。Kali Linuxs是在Debian项目基础上构建的，并添加了超过 300 款特殊用途的软件包，所有这些软件包和信息安全相关，特别是渗透测试领域的软件包。

Debian 是一个具有多种不同版本的免费操作系统软件，我们通常使用“发行版（distribution）”来表示一个特定版本，如 Debian 稳定版或 Debian 测试版。Kali Linux 也是一样的，例如它也有 Kali 滚动发行版（Rolling）。

1 https://www.kali.org
2 https://www.debian.org

1.1 简要回顾

Kali Linux项目是在 2012 年悄悄启动的，当时Offensive Security公司决定用一个新项目，来替代享誉很高但需要手动维护的BackTrack Linux项目。这个新项目是一个在Debian基础架构和改进软件包管理技术基础上构成的Debian衍生版本 [1]。他们最终决定在Debian发行版上构建Kali，是因为Debian以其质量、稳定性和丰富可选的应用程序库而闻名于世。这也是为什么我（Raphaël）作为Debian顾问参与这个项目的原因。

2013 年 3 月，也就是 1 年以后，第一个 Kali 版本（1.0 版）发布。这个版本基于 Debian 7“Wheezy”——当时 Debian 最稳定的发行版。在开发的第一年中，我们打包了数百种与渗透测试相关的应用程序，构建了 Kali 的基础架构。虽然集成的工具数量庞大，但这个清单是经过精心策划的，而且丢弃了已不再维护的工具，或在更好的工具中已包含的重复功能。

在 1.0 版发布后的两年内，得益于 Linux 内核的更新，Kali 发布了许多增量更新，扩展了收录应用程序的范围，并改进了硬件支持。通过对持续集成技术的投入，我们确保所有重要的软件包都处于可安装状态，并且始终可以创建定制自生镜像。

Debian 8“Jessie”问世时，我们努力在它之上重建 Kali Linux。在这个新版本中，我们决定采用并增强在 Kali Linux 1.x 版本中未采用的 GNOME Shell（1.x 中使用 GNOME Fallback）：我们添加了一些 GNOME Shell 扩展来填补缺失功能，特别是经典的“应用程序”菜单。该项工作的成果就是 2015 年 8 月发布的 Kali Linux 2.0。

GNOME 是 Kali Linux 的默认桌面环境

桌面环境是共享同一个图形工具包的图形界面应用程序的集合。用户的工作站经常会采用图形界面，而服务器一般不使用。常见的桌面环境会提供一个应用启动器、一个文件管理器、一个 Web 浏览器、一个 Email 客户端以及一套办公软件等。

GNOME[2]是最为流行的桌面环境之一（其他还有KDE[3]、Xfce[4]、LXDE[5]、MATE[6]）并且能够通过Kali Linux主ISO镜像安装。如果你不喜欢GNOME，可以轻松地使用你喜欢的桌面环境，重新编译一个自定义的ISO镜像。本书的第 9 章“高级用法”会告诉你如何操作。

1 https://wiki.debian.org/Derivatives/Census
2 https://www.gnome.org
3 https://www.kde.org
4 http://www.xfce.org
5 http://lxde.org
6 http://mate-desktop.org

与此同时，我们付出更多努力，以确保 Kali Linux 始终拥有所有渗透测试工具的最新版。但不幸的是，这个目标与使用 Debian 稳定版（Debian Stable）作为发布基础有点冲突，我们发现需要制作许多旧版本应用程序的补丁包。这是因为 Debian 稳定版会优先考虑软件稳定性，这经常会导致将新版本应用集成到操作系统中需要相当长的时间。出于在持续集成方面的成本投入考虑，在 Debian 测试版（Debian Testing）之上重建 Kali Linux 是一个很合理的做法，这使得我们能够在第一时间使用所有 Debian 软件包的最新版。Debian 测试版具有更加激进的更新周期，这更符合 Kali Linux 的理念。

这实际上就是Kali滚动发行版（Kali Rolling）的概念 [1]。虽然滚动发行版已经应用相当一段时间了，但Kali 2016.1 是第一个官方的滚动发行版：当你安装最新的Kali版本时，你的系统会跟踪Kali滚动发行版的软件库，并且每天都会收到更新补丁。而在此之前的Kali版本实际上可以被看作是在标准Debian发行版快照中注入了Kali软件包。

使用滚动发行版有许多好处，但无论是对于我们这些正在构建发行版的开发人员，还是对于需要应对频繁更新并对兼容性进行反复尝试的用户而言，它也带来了许多挑战。而本书为你提供了使用 Kali Linux 的过程中应对这些挑战所需要的一切知识。

1.2　和Debian的关系

当前的Kali Linux发行版基于Debian测试版 [2]。因此，Kali Linux中的大多数可应用软件包都可以直接从Debian软件库中获取。

虽然 Kali Linux 重度依赖 Debian，但从某种意义上说，它也是完全独立的，因为我们拥有自己的操作系统基础设施，并且能够自由更改它。

1.2.1　软件包处理流程

在 Debian 方面，贡献者每天都在更新软件包并将其上传到 Debian 非稳定版（Debian Unstable）。一旦所有重大问题在非稳定版中被解决，软件包就被迁移到 Debian 测试版。这

1 译者注：与滚动发行版（rolling distribution）相对的是固定发行版（fixed release），后者每隔固定时间发布一个新版镜像，并同时增加版本号（如 Ubuntu 14.04 升级到 16.04），而滚动发行版随时都会更新，下载的原始镜像的版本号通常由生成镜像的时间确定（如 2017.1）。固定发行版通常经过严格的测试，性能比较稳定，但软件更新周期很长；而滚动发行版由于无法充分测试，因此稳定性欠佳，但可以提供最新的软件版本。通常采用滚动发行版的操作系统有 Arch Linux 和 Kali Linux。

2 https://www.debian.org/releases/testing/

个迁移过程同时还要确保没有造成任何 Debian 测试版已存在依赖文件被破坏。我们的目标是使 Debian 测试版始终处于可用（甚至是可发布）状态。

Debian 测试版目标与 Kali Linux 的目标相当一致，因此我们选择它作为基础。要在发行版本中添加 Kali 特定软件包，需要分两步走。

首先，强制向 Debian 测试版注入 Kali 软件包（位于 kali-dev-only 库中）来构建 kali-dev 库。这个库会时不时地出问题。比如，Kali 的软件包在使用较新库文件重新编译之前可能无法安装。在一些情况下，还需要想办法去处理那些 fork 的软件包，要么想办法让新版软件包能够顺利安装，要么修改依赖于此 fork 软件包的另一个软件包安装程序，以便能够顺利安装。无论如何，kali-dev 都不是给最终用户准备的。

与 Debian 测试版基于 Debian Unstable 构建如出一辙，kali-rolling 滚动发行版是基于 kali-dev 构建的，其也是推荐 Kali Linux 用户跟踪使用的版本。只有当所有的依赖问题都解决后，软件包才会被迁移到 kali-rolling 滚动发行版中。

1.2.2 同Debian的区别

作为一种设计决策，尽量减少 fork 软件包的数量。然而，为了实现 Kali 的一些独特功能，我们不得不对 Debian 做一些修改。为了降低这些变化所造成的影响，我们努力将这些需求转移到上游，可以通过直接集成这些功能，或添加一些钩子程序，来避免进一步修改上游程序包。

Kali Package Tracker[1]帮助追踪我们与Debian的分歧。我们可以随时查看哪个包已被fork分支，它是否与Debian同步，或者是否需要更新。所有软件包都保存在同时托管着Debian分支和Kali分支的Git库 [2]中。基于此，更新fork包变成简单的两步过程：更新Debian分支，然后将其合并到Kali分支。

虽然Kali的fork包数量相对较少，但附加软件包数量却相当大，截至 2017 年 4 月有近 400 个。这些软件包中的大多数都是符合“Debian自由软件指南”[3]政策的免费软件，我们最终目标是尽可能在Debian中维护这些软件包。这就是为什么我们要努力遵守Debian政策条款 [4]，并遵循那些在Debian中使用良好的包管理经验的原因。不幸的是，仍然存在很多例外情况，在这些情况下，几乎无法创建合适的软件包。由于时间紧迫，几乎没有多少软件包被推送给Debian。

1 https://pkg.kali.org/derivative/kali-dev
2 http://git.kali.org
3 https://www.debian.org/social_contract
4 https://www.debian.org/doc/debian-policy/

1.3　设计目标和使用场景

虽然 Kali 的设计重点可以快速归纳为“渗透测试和安全审计”，但在这些活动背后实际上涉及了许多不同的任务。Kali Linux 的定位是一个框架，因为它包含了许多用途各异的工具（尽管它们在渗透测试中可能会被组合使用）。

例如，Kali Linux 可以用于各种类型的计算机。除了用于最常见的渗透测试工程师的便携式电脑、取证分析师的工作站外，还可以被希望对网络进行监控的系统管理员安装在服务器上，等等。你可能想象不到，它还可以安装在使用 ARM CPU 的随身嵌入式设备上，这些设备可以放置在无线网络范围内或者插入目标用户计算机中。许多 ARM 设备由于其体积小、功耗低，天生就是完美的攻击利器。此外，Kali Linux 还可以部署在云端，以迅速建立一个密码破解机器集群，或者安装在手机和平板电脑上进行真正的便携式渗透测试。

但这还不是全部，渗透测试人员还需要服务器，用于在渗透测试团队中使用协作软件、设置一个用于网络钓鱼的 Web 站点、运行漏洞扫描工具以及其他相关活动等。

一旦启动了 Kali，你很快会发现，Kali Linux 的主菜单是按照测试人员和其他信息安全专业人员相关的各种任务与活动来组织的，如图 1.1 所示。

图1.1　Kali Linux的应用程序主菜单

其中包括这些任务与活动。

- **信息收集：**收集目标网络及其结构的数据，识别计算机、操作系统和其上运行的各种服务。识别信息系统中潜在敏感部分。从正在运行的目录服务中提取各种清单。
- **漏洞分析：**快速测试本地或远程系统是否受到一些已知漏洞或者不安全配置的影响。漏洞扫描程序使用包含大量漏洞特征的数据库，来识别潜在的漏洞。
- **Web 应用程序分析：**识别 Web 应用程序中的错误配置和安全漏洞。由于这些应用程序通常都是完全开放给公众访问的，因此也成为了攻击者的理想目标，识别和修补这些漏洞至关重要。
- **数据库评估：**从 SQL 注入到攻击身份认证凭据，数据库攻击是十分常见的攻击向量。可以从这里找到从 SQL 注入到数据提取分析的各类攻击工具。
- **密码攻击：**身份认证系统始终是一种可行的攻击向量。从在线密码攻击工具到攻击加密或哈希的离线工具，大量工具都被集成在这里。
- **无线攻击：**无线网络易于被接触的特性意味着它们是一种常见的攻击向量。由于 Kali 广泛支持多种无线网卡，所以它无疑是对各类无线网络实施攻击最为明智的选择。
- **逆向工程：**逆向工程是可以支持多种目标的活动。在攻击方面，它是识别漏洞和漏洞利用程序开发的主要方法之一。在防御方面，它能够用于分析在攻击中使用的恶意软件，在这个方面，逆向工程的目标是要识别出给定代码片段的功能，使用逆向工程工具提供的能力，Kali 可以用于识别恶意软件。
- **漏洞利用工具：**利用先前已识别的漏洞，可以使你得到远程机器（或设备）的访问权限。还有可能在初步访问权限基础上使用权限提升等各种攻击方式，在这台已被攻陷的本地计算机上或本地网络其他可访问的计算机上获取更高权限。该类别里面包含了一些工具，可以帮助你简化开发漏洞利用工具的过程。
- **嗅探和欺骗：**在网络中畅游的同时获得直达数据的入口，常常是攻击者非常愿意干的事。在这里，可以找到能够允许模拟合法用户的欺骗工具，以及能够直接捕获并分析网络数据的嗅探工具。当这些工具被组合起来使用时会非常强大。
- **后渗透：**一旦你找到通往系统的入口，常常会希望能够保持住这个访问入口，或者进一步扩大在内网的控制面。可以在这里找到相关的工具来实现这些目标。
- **取证：**取证用的 Linux 自生系统环境近些年来非常流行。Kali 包含了大量基于 Linux 的常用取证工具，这些工具能够帮助你完成从初步分析到数据镜像再到完整数据分析，以及案例管理等全部数据取证流程。
- **报告工具：**渗透测试项目通常以提交一份包含了所有发现问题的报告而宣告结束。此类别里面包含的工具可以帮助整理信息收集工具所收集到的数据，发现隐含的关系以

及整合各种报告内容等。

- **社会工程工具：**如果攻击目标的技术防护工作做得很到位，则对人性弱点的利用往往会成为比较容易实施的攻击向量。只要施加适当的影响，人们可能就会被诱使去做一些无视安全规则的事情。秘书刚刚插入的 U 盘中是一个无害 PDF 文件，还是会给电脑安装后门程序的特洛伊木马？会计刚登录的银行网站是他想要登录的那个，还是用于网络钓鱼的完美假冒站点？这个类别中包含了帮助实现此类攻击的工具。
- **系统服务：**此类别中的工具用来方便你启动和停止那些作为系统服务后台运行的应用程序。

1.4　Kali Linux的主要功能

Kali Linux 是一个 Linux 发行版，其中包含了数百种为目标用户（渗透测试人员和其他安全专业人员）量身定做的软件工具集合。它还附带一个安装程序，可以将 Kali Linux 安装为计算机上的主要操作系统。

虽然与其他现有 Linux 发行版十分类似，但仍然有一些特性可以将 Kali Linux 与其他 Linux 发行版区别开来，其中许多功能是针对渗透测试人员的特定需求量身定做的。下面我们来看看其中的一些功能。

1.4.1　一个自生系统

与大多数 Linux 发行版相同，你下载的主 ISO 镜像不仅能够帮助安装操作系统，同时也可作为可自启动的自生系统。换句话说，可以在不安装 Kali Linux 的情况下使用它，而只需启动 ISO 镜像（通常在将镜像复制到 U 盘之后）即可。

这套自生系统包含渗透测试人员最常用的工具，因此即便你日常使用的操作系统不是 Kali Linux，也只需插入移动硬盘或 U 盘并重新启动来运行它。但是，请注意，在默认配置下不会保留你对系统做出的更改，重新启动之后这些更改会丢失。如果你使用 U 盘而且配置了持久化选项（参见 9.4 节的内容），才可以根据自己喜好对系统进行调整（例如修改配置文件、保存报告、升级软件以及安装其他软件包等），这些更改将在重新启动后保留。

1.4.2 取证模式

一般来说，在对系统进行取证工作时，希望避免发生任何会修改被分析系统数据的行为。然而现代桌面操作系统都会尝试自动挂载其检测到的所有磁盘，这和我们的目标是违背的。为了避免此类情况发生，Kali Linux 有一个可以从引导菜单启用的取证模式：它将禁用所有这些功能。

自生系统用于取证目的时特别有用，因为可以使用 Kali Linux 启动任何一台计算机，而无须访问或修改其硬盘。

1.4.3 一个自定义的Linux内核

Kali Linux 总是提供一个基于 Debian Unstable 版本定制的新版 Linux 内核。这确保了硬件支持的稳定性，对于支持型号众多的无线设备而言非常重要。这个版本的内核增加了无线注入支持补丁，因为许多无线安全评估工具都依赖于此功能。

由于许多硬件设备需要最新的固件文件（可以在/lib/firmware/中找到），所以默认情况下，Kali 会将它们全部安装——包括 Debian 提供的非免费（no-free）部分固件。在 Debian 中这些固件在默认情况下是不安装的，因为它们未开放源码，并非 Debian 的固有部分。

1.4.4 完全可定制

Kali Linux 是由渗透测试人员创建的，也是为渗透测试人员创建的。但我们知道，并非所有人都认同我们的设计决策和工具选择。考虑到这一点，我们始终确保 Kali Linux 可以根据你自己的需求和喜好轻松进行定制。为此，我们发布用于构建官方 Kali 镜像的 live-build 配置，以便你可以根据自己的喜好进行自定义。得益于 live-build 的强大功能，你很容易就能上手对系统做出各种更改。

live-build 包含许多功能，如修改已安装系统，安装补充文件，安装附加软件包，运行任意命令，以及修改用于 debconf 的预设安装问题答案等。

1.4.5 一个值得信任的操作系统

作为一个用于安全测试的 Linux 发行版，应当让用户有充足的理由信任它，它需要得到

公众的监督，而且任何人随时都可以审查它的源代码。Kali Linux 由一个小而精的团队开发，工作流程透明并遵循最佳安全实践：他们上传了签名的源码包，然后由专用的构建守护进程将其编译为软件包。这些软件包之后会被校验，并作为签名程序库的一部分进行发布。

这些针对软件包的所有工作都可以在构建Kali源码的Git仓库 [1]中（含有签名标签）完整地查看。每个软件包的演进过程也可以通过Kali软件包跟踪器 [2]来进行追踪。

1.4.6　可以在众多的ARM设备中使用

Kali Linux 为 armel、armhf 和 arm64 等 ARM 架构提供二进制包。得益于 Offensive Security 提供的易于安装的镜像文件，Kali Linux 可以部署在许多不同的设备上，例如智能手机、平板电脑、Wi-Fi 路由器和不同形状大小的计算机上。

1.5　Kali Linux的设计策略

虽然 Kali Linux 尽可能地遵循 Debian 政策，但由于安全从业人员的特殊需求，我们在某些方面做出了一些明显不同的设计选择。

1.5.1　默认使用root用户

大多数 Linux 发行版鼓励在运行系统时使用非特权账户，并在需要管理权限时使用诸如 sudo 之类的实用程序。这是合理有效的安全建议，在用户和任何具有潜在破坏性或毁灭性的操作系统操作之间提供了额外的保护屏障。这对于多用户系统尤其如此，用户权限分离是必要的，一个用户的不当行为可能会破坏或摧毁许多用户的工作。

由于 Kali Linux 中包含的许多工具只能通过特权执行，所以 Kali 的默认用户账户是 root。与其他 Linux 发行版不同，安装 Kali 时不会提示你创建非特权用户。这个特殊政策与大多数 Linux 系统大相径庭，并且容易让经验不足的用户感到困惑。使用 Kali 时，初学者应特别小心，因为大多数毁灭性的错误都发生在 root 权限下的操作。

1 http://git.kali.org
2 http://pkg.kali.org

1.5.2 禁用了网络服务

与 Debian 相反，Kali Linux 在默认情况下会禁用任何能够在公共网络接口上进行侦听的已安装服务，例如 HTTP 和 SSH。

这一决定背后的逻辑依据是，由于意外的网络交互可能会宣示你的存在，从而增加被察觉的风险，而这对测试活动是非常不利的，因此在测试过程中必须尽可能地减少曝光。

你仍然可以通过运行 systemctl enable service 手动启动需要的任何服务。本书后面的第 5 章对此会做详细介绍。

1.5.3 一个精心组织的应用集合

Debian 的目标是成为通用的操作系统，它对于收录哪些软件包并没有过多的限制，只要每个软件包有一个维护者就好。

作为对比，Kali Linux 并不打包每个可用的渗透测试工具。相反，我们的目标是提供一个能够涵盖渗透测试人员大多数日常工作的最佳免费授权工具集合。

Kali 开发团队中的渗透测试工程师们是这一甄选过程的主要决策者，我们认为他们通过经验和专业知识做出了明智的选择。在大多数情况下的确如此，但仍然会存在由于个人偏好不同而难以找到合适工具软件的情况。

以下是我们评估新工具软件时考虑的一些要点：

- 渗透测试整个流程中工具软件的有用程度。
- 工具软件是否具有独特的功能。
- 工具软件的使用许可。
- 工具软件对资源的要求。

维护一个持续更新并且好用的渗透测试工具库是一项具有挑战性的任务。我们欢迎大家在Kali Bug 跟踪[1]中的“新工具请求”（New Tool Requests）类别中提出建议。应当描述清楚工具有哪些独特的功能、与其他类似工具的比较等，这样才更有助于新的工具被开发组采纳。

1 http://bugs.kali.org

1.6　小结

在本章中，我们介绍了 Kali Linux，包括 Kali 的历史和主要功能，并提供了几个用例。还讨论了在开发 Kali Linux 时采用的一些策略。

要点提示：

- Kali Linux[1]是基于Debian GNU/Linux的企业级安全审计Linux发行版。Kali的目标用户是安全专业人员和IT管理员，主要用于进行高级渗透测试、取证分析和安全审计。
- 与大多数主流操作系统不同，Kali Linux 是滚动发行版本，这意味着你每天都会收到更新。
- Kali Linux发行版基于Debian 测试版 [2]。因此，Kali Linux中的大多数软件包都可以直接从Debian存储库中获取。
- 虽然 Kali 的主要任务可以简单归纳为“渗透测试和安全审计”，但它还有很多不同的用途，包括网络监控、取证分析、无线网监控等，而且能够安装在多种嵌入式设备和移动平台上。
- 可以从 Kali 主菜单方便地找到各种任务所需的工具，包括漏洞分析、Web 应用分析、数据库评估、密码攻击、无线攻击、逆向工程、开发工具、嗅探和欺骗、后渗透工具、取证、报告工具、社会工程工具和系统服务等。
- Kali Linux 具有许多先进的功能，包括用作自生（非安装自启动）系统、稳健安全的取证模式、定制的 Linux 内核、完全自定义的能力、可靠安全的基础操作系统、ARM 架构平台上的安装能力、安全的默认网络策略等。在下一章中，我们将以自生模式实际体验 Kali Linux。

练习题

练习1——搭建环境

1．在 VMWare 中创建一个新的虚拟机（类型设置为 Debian 64 位）。

1 https://www.kali.org
2 https://www.debian.org/releases/testing

2．为它分配至少 2GB 内存、2 个 CPU 和 30GB 硬盘。
3．将 Kali 的 ISO 文件挂载到虚拟 CDROM 上。
4．确认虚拟机网络设置为 NAT 模式。
5．启动虚拟机，观察并理解 Kali 的启动选项。

思考题

1．Kali 1.0、2.0 和滚动发行（Rolling）版分别是基于 Debian 的什么版本开发的？
2．Kali 的 Live 自生实例和安装实例的主要区别是什么？
3．自生模式（Live）和取证模式（Forensics）的区别是什么？
4．怎样确认取证模式（Forensics）工作正常？
5．将工具包含到 Kali 中最好的途径是什么？
6．举出一些 Kali 中的功能亮点（cool features）。

第2章　Kali入门

内容

- 下载 Kali ISO 镜像
- 用自生模式启动 Kali ISO 镜像
- 小结

与其他一些操作系统不同，由于 Kali Linux 的磁盘镜像是一个自生 ISO，因此上手非常简单，这意味着可以直接启动下载的镜像，而不需要执行任何安装步骤。这也意味着可以将同一个镜像文件用于渗透测试、取证，用作可引导的 USB 或 DVD-ROM 镜像，或者用它把 Kali 持久安装在物理或虚拟硬件上。

由于 Kali 的这种简便易用性，很容易让人忘记必须采取某些防御措施。Kali 用户的测试目标经常是一些别有用心的人，包括那些受国家资助的团体、犯罪组织的成员，以及个人黑客等。Kali Linux 开放源码的特性，使得构建和发行伪造版本变得相对容易，因此，必须从官方来源下载，并且要验证下载的完整性和真实性。这对于经常访问敏感网络并受客户之托能够接触到敏感数据的安全专业人士而言尤为重要。

2.1　下载一个Kali ISO镜像

2.1.1　从哪里下载

Kali Linux ISO 镜像唯一的官方来源是 Kali 网站的“下载”栏目。由于 Kali 广受欢迎，许多网站都提供 Kali 镜像用于下载，但这些网站不可信赖，因为这些未知来源的镜像确实存在被恶意软件感染或其他的风险，这有可能会对系统造成无法弥补的损害。

➨ `https://www.kali.org/downloads/`

该网站通过 HTTPS 方式提供，难以被仿冒。攻击者仅仅实施中间人攻击是不够的，他

还需要由被攻击者信任的 Transport Layer Security (TLS)认证机构签署的证书。认证机构之所以存在，正是为了防范此类问题。他们只向已验证身份并拥有网站所有权证据的人员发放证书。

cdimage.kali.org 在下载页面上找到指向 cdimage.kali.org 域名的链接，你会被引导到一个较近的镜像站点上，以便获得更快下载速度，也能够减轻 Kali 网站中心服务器的压力。

可以在这里找到一个包含所有可用镜像的列表：

➨ `http://cdimage.kali.org/README. mirrorlist`

2.1.2 下载什么内容

官方下载页面显示了 ISO 镜像简短列表，如图 2.1 所示。

Download Kali Linux Images

We generate fresh Kali Linux image files every few months, which we make available for download. This page provides the links to **download Kali Linux** in its latest official release. For a release history, check our Kali Linux Releases page. Please note: You can find unofficial, untested weekly releases at http://cdimage.kali.org/kali-weekly/.

Image Name	Download	Size	Version	sha256sum
Kali 64 bit	ISO | Torrent	2.6G	2017.1	49b1c5769b909220060dc4c0e11ae09d97a270a80d259e05773101df62e11e9d
Kali 32 bit	ISO | Torrent	2.7G	2017.1	501b3747e5ac7c698217392fe49ec21dacee277404500fc49d4a0ee82625aabe
Kali 64 bit Light	ISO | Torrent	0.8G	2017.1	5c0f6300bf9842b724df92cb20e4637f4561ffc03029cdcb21af3902442ae9b0
Kali 32 bit Light	ISO | Torrent	0.8G	2017.1	6c83101ecf8702c7d93d32562e822b639d5c577314b448e3b8330995e0f07e0f
Kali 64 bit e17	ISO | Torrent	2.4G	2017.1	ae293cf679f38a4f17d090a272ccb13d7619e66d4502374154186c12891fb99c
Kali 64 bit KDE	ISO | Torrent	2.7G	2017.1	839741fec378114ff068df3ec2dbed9d8e4fae613e690d50b25ce9cc1468104b
Kali 64 bit Mate	ISO | Torrent	2.6G	2017.1	3ea748aa8c5f50d80f020acdbca5f0398ee90242bb4413c12985e1865186ca9e
Kali 64 bit Xfce	ISO | Torrent	2.5G	2017.1	8a17c2f54850585760b9d32a22e26df9a28f395b401753fa0a9b298aef4c4593
Kali 64 bit LXDE	ISO | Torrent	2.5G	2017.1	35eae65aaaabba8188dfd963e45b7b4d76e0684e7721c7d232cf18320b7cae3b
Kali armhf	Image | Torrent	0.5G	2017.1	a75199aa8a3d7b64561bc03fcd6e3ff8b94743c8769eecfaa4b719f04f7cbb63
Kali armel	Image | Torrent	0.4G	2017.1	180414422196f0797c1ea5f3c18682bc4b3ced871cb3e874e90de52dd4af877c

图2.1 提供下载的镜像列表

所有标有 32 位或 64 位的磁盘镜像适用于大多数主流台式机和笔记本电脑 CPU。如果使用一台主流计算机下载，那么它很可能使用了 64 位处理器。如果不确定处理器的类型，请放心，所有 64 位处理器都可以运行 32 位指令。你可以随时下载并运行 32 位镜像。然而，反过来是不行的，32 位处理器无法运行 64 位版本。有关详细信息，请参阅下栏的内容。

如果打算在嵌入式设备、智能手机、Chromebook、无线路由器或任何其他具有 ARM 处理器的设备上安装 Kali，则必须使用 Linux armel 或 armhf 镜像。

CPU 是 32 位还是 64 位？

在 Windows 下，运行“系统信息”应用程序（在“附件”→“系统工具”中可以找到）来查看 CPU 信息。在系统概述屏幕，可以检索“系统类型”字段，如果是 64 位 CPU，会包含“基于 x64 的 PC”字样。如果是 32 位 CPU，会包含“基于 x86 的 PC”字样。

在 OS X/macOS 下，没有一个标准的应用能够显示这一信息，不过仍然可以通过从终端中运行 uname –m 命令的输出看到这些信息。它会在运行 64 位内核的计算机上返回 x86_64 字样，在运行 32 位内核的计算机上返回 i386 或其他类似的字样（i486、i586 或 i686）。

在 Linux 下，可以查看/proc/cpuinfo 虚拟文件中的 flag 字段。如果它包含 lm 属性，那么你的 CPU 是 64 位的，否则是 32 位的。下面的命令能够告诉你 CPU 的类型：

```
$ grep -qP '^flags\s*:.*\blm\b' /proc/cpuinfo &&
    ➥ echo 64-bit || echo 32-bit
64-bit
```

现在我们知道了需要 32 位还是 64 位镜像，只剩一个步骤了：选择镜像种类。默认 Kali Linux 镜像和 Kali Linux Light 版本都可以用于运行自生系统或作为启动安装程序的自生 ISO。它们的区别仅仅在于预装程序的数量。默认镜像提供 GNOME 桌面，并提供了大量软件包供大多数渗透测试人员使用。light 版本使用 Xfce 桌面（其对于系统资源的要求要低得多），提供数量有限的软件包，并且允许你只选择需要的应用程序。剩下的镜像版本使用了其他桌面环境，但与主要镜像一样拥有大量软件包。

确定了所需镜像后，可以通过单击相应行中“ISO”链接来下载镜像。或者，可以通过单击“Torrent”链接从 BitTorrent 点对点网络下载镜像，前提是你已有一个与.torrent 扩展名关联的 BitTorrent 客户端。

在下载 ISO 镜像的过程中，请注意在 sha256sum 列中写入的校验值。镜像下载完成后，请用此校验值来验证下载镜像是否与 Kali 开发团队在线放置的镜像一致（请参阅下一节内容）。

2.1.3 验证完整性和真实性

安全专业人员必须验证其工具的完整性，这一方面可以保障数据和网络安全，同时也确保客户安全。虽然 Kali 下载页面受到 TLS 保护，但下载链接指向未加密、不会对潜在中间人攻击提供保护的 URL。Kali 需要依赖外部镜像网络来分发镜像，因此不应该盲目信任下载的内容。你跳转到的镜像站点可能已经被攻陷，或者你本人就是攻击的受害者。

为了缓解这一问题，Kali 项目总是提供其发行镜像的校验值。但要使这样的检查有效，必须确保你拿到的校验值确实是 Kali Linux 开发团队所发布的。有多种方法可以来确认。

依靠TLS保护的网站

当从受 TLS 保护的下载页面中获得校验值时，它的来源真实性间接由 X.509 证书安全模型保证：你看到的内容来自一个 TLS 证书申请者控制之下的网站。

现在生成下载镜像的校验值，并确保它与你从 Kali 网站记录的内容相符：

```
$ sha256sum kali-linux-2017.1-amd64.iso
49b1c5769b909220060dc4c0e11ae09d97a270a80d259e05773101df62e11e9d  kali-
    ➥ linux-2016.2-amd64.iso
```

如果生成的校验值与 Kali Linux 下载页面匹配，那么即获取了正确文件。如果校验值不同就有问题了。不过这并不一定代表镜像被人为损害或攻击，因为下载的文件偶尔会因为网络原因而被损坏。如果可能，请再次尝试从另一个官方 Kali 镜像站点下载（有关可用镜像的更多信息，请参阅“cdimage.kali.org”）。

依靠PGP的信任网

你可能会不信任 HTTPS 进行的身份验证，这虽然有点偏执，但也并非杞人忧天。现在有很多管理不善的证书颁发机构签发流氓证书而最终被滥用的例子。你也可能是“友好的”中间人攻击的受害者，这种情况通常是由一些网络监控设备造成的，这类设备在许多公司内部网络上很常见，它通常使用一个自定义的浏览器信任模型，并为加密网站提供一个假证书，从而允许公司审计人员对加密网络流量进行监控。

针对这种情况，我们还提供了一个 GnuPG 密钥，用来对所提供的镜像校验值进行签名。密钥标识符及指纹如下所示：

```
pub  rsa4096/0xED444FF07D8D0BF6 2012-03-05 [SC][expires: 2018-02-02]
    Key fingerprint = 44C6 513A 8E4F B3D3 0875 F758 ED44 4FF0 7D8D 0BF6
uid                  [ full ] Kali Linux Repository <devel@kali.org>
```

```
sub  rsa4096/0xA8373E18FC0D0DCB 2012-03-05 [E][expires: 2018-02-02]
```

这个密钥是全球信任网络的一部分，因为至少我（Raphaël Hertzog）已经签署了。而且作为 Debian 开发人员的我是 GnuPG 重度使用者，所以我本人就是该信任网络的一分子。

PGP/GPG 安全模式非常独特。任何人都可以生成具有任何身份的密钥，但是你只能信任那些已经通过信任者另一个密钥签名的密钥。当你签署密钥时，你要确定遇到的密钥持有人和相关联身份是正确的。初始密钥集由你自己定义，很明显它会包含你自己的密钥。

这种模式有其自身局限性，因此可以选择通过 HTTPS（或从密钥服务器）下载 Kali 公钥文件，并且决定信任它。因为它的指纹信息与我们在许多地方（包括本书）声明的内容相符。

```
$ wget -q -O - https://www.kali.org/archive-key.asc | gpg --import
[ or ]
$ gpg --keyserver hkp://keys.gnupg.net --recv-key ED444FF07D8D0BF6
gpg: key 0xED444FF07D8D0BF6: public key ″Kali Linux Repository
     ➥ <devel@kali.org>″ imported
gpg: Total number processed:1
gpg:               imported:1 (RSA: 1)
[...]
$ gpg --fingerprint 7D8D0BF6
[...]
     Key fingerprint = 44C6 513A 8E4F B3D3 0875 F758 ED44 4FF0 7D8D 0BF6
[...]
```

当你得到密钥后，可以使用它来验证下载镜像校验值。接下来我们用校验值（SHA256SUMS）和相关的签名文件（SHA256SUMS.gpg）来下载文件并验证签名：

```
$ wget http://cdimage.kali.org/current/SHA256SUMS
[...]
$ wget http://cdimage.kali.org/current/SHA256SUMS.gpg
[...]
$ gpg --verify SHA256SUMS.gpg SHA256SUMS
gpg: Signature made Thu 16 Mar 2017 08:55:45 AM MDT
gpg:                using RSA key ED444FF07D8D0BF6
gpg: Good signature from ″Kali Linux Repository <devel@kali.org>″
```

如看到“正确签名”（Good signature）这一消息，即可信任 SHA256SUMS 文件内容，并使用它来验证下载文件。否则就有问题，这时候应该检查是否是从 Kali Linux 合法镜像站点下载的文件。

请注意，可以使用以下命令验证下载文件是否具有与 SHA256SUMS 中列出的相同校验

值，前提是需要在下载 ISO 文件相同的目录中运行它：

```
$ grep kali-linux-2017.1-amd64.iso SHA256SUMS | sha256sum -c
kali-linux-2017.1-amd64.iso: OK
```

如果命令没有响应，则意味着下载文件与 Kali 团队发布文件不同。请不要信任这个文件，更不要使用它。

2.1.4 将镜像复制到DVD-ROM或USB驱动器中

除非想在虚拟机中运行 Kali Linux，否则 ISO 镜像文件本身应用范围有限。如果你想在你的计算机上启动 Kali Linux，必须将其刻录在 DVD-ROM 上，或将其复制到 USB 驱动器上。

由于每个人使用的平台和环境不同，这里不会介绍如何将 ISO 镜像刻录到 DVD-ROM 上。但在大多数情况下，右键单击.iso 文件会呈现一个上下文菜单，通过该菜单可以启动一个 DVD-ROM 刻录应用程序。可以在自己的计算机上试试看！

警告

在这一节，将学习如何将 Kali Linux ISO 镜像写到一块磁盘上。在开始前一定要反复检查这个目标磁盘是不是你想要写入的磁盘，因为这个操作会完全擦除目标磁盘上的数据，并有可能造成无法恢复的损失。

在Windows上创建一个能够引导的Kali USB驱动器

作为先决条件，需要下载并安装 Win32 Disk Imager：

➨ `https://sourceforge.net/projects/win32diskimager/`

将 U 盘插入一台安装了 Windows 的计算机中，并记下与之相关的驱动器指示符（例如“E：\”）。

启动 Win32 Disk Imager 并选择要在 U 盘上复制的 Kali Linux ISO 文件。验证所选设备的驱动器符号是否与分配给 U 盘的驱动器相符。确定选择了正确的驱动器后，单击 Write 按钮，并确认覆盖 USB 驱动器的内容，如图 2.2 所示。

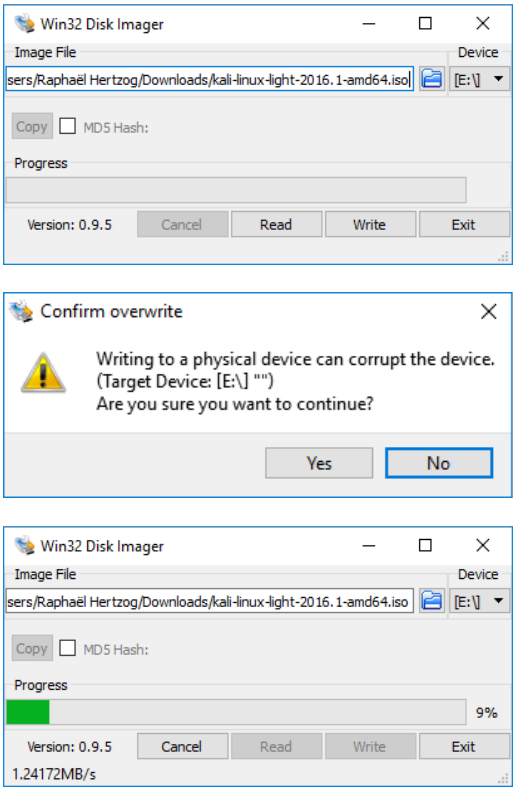


图2.2　Win32 Disk Imager正在工作

复制完成后，从 Windows 系统安全地弹出 USB 驱动器。现在便可以使用 U 盘启动 Kali Linux 了。

在Linux系统上创建一个能够引导的Kali USB驱动器

在 Linux 环境中创建可启动的 Kali Linux U 盘很简单。在许多 Linux 发行版中默认安装的 GNOME 桌面环境附带一个 Disks 实用程序（在 gnome-disk-utility 包中，已经安装在 Kali 镜像中）。该程序会显示当前的磁盘列表，当插入或拔出磁盘时，它将动态刷新。当你在磁盘列表中选择 USB 密钥时，将会出现详细的提示信息帮助你确认选择正确的磁盘。可以在标题栏中找到该设备名称，如图 2.3 所示。

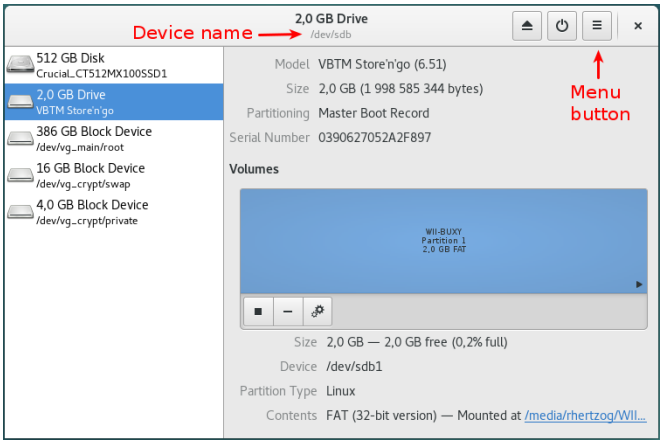


图2.3　GNOME磁盘

单击菜单按钮，然后在显示的弹出菜单中选择恢复磁盘镜像（Restore Disk Image…）。选择之前下载的 ISO 镜像，然后单击 Start Restoring...（开始恢复）按钮，如图 2.4 所示。

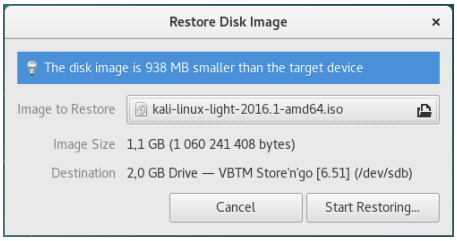


图2.4　恢复磁盘镜像对话框

当开始在 USB 驱动器上复制镜像后，可以坐下来享用一杯咖啡了。如图 2.5 所示的窗口显示了磁盘镜像恢复进度。

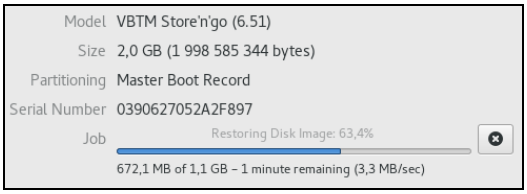


图2.5　镜像恢复进度

从命令行创建启动 U 盘

虽然使用图形界面制作启动 U 盘很方便，不过习惯使用命令行的用户也可以通过简单操作完成这一过程。

当插入一个 USB 驱动器时，Linux 内核会检测到它并给其分配一个名字，并记录在内核运行日志中。可使用 dmesg 命令找到它。

```
$ dmesg
[...]
[234743.896134] usb 1-1.2: new high-speed USB device number 6
     ➥ using ehci-pci
[234743.990764] usb 1-1.2: New USB device found, idVendor=08ec,
     ➥ idProduct=0020
[234743.990771] usb 1-1.2: New USB device strings: Mfr=1,
     ➥ Product=2, SerialNumber=3 [234743.990774] usb 1-1.2:
     ➥ Product: Store'n'go
[234743.990777] usb 1-1.2: Manufacturer: Verbatim
[234743.990780] usb 1-1.2: SerialNumber: 0390627052A2F897
[234743.991845] usb-storage 1-1.2:1.0: USB Mass Storage device
     ➥ detected
[234743.992017] scsi host7: usb-storage 1-1.2:1.0
[234744.993818] scsi 7:0:0:0: Direct-Access VBTM Store'n'go
     ➥ 6.51 PQ:0ANSI:0CCS
[234744.994425] sd 7:0:0:0: Attached scsi generic sg1 type 0
[234744.995753] sd 7:0:0:0: [sdb] 3903487 512-byte logical
     ➥ blocks: (2.00GB /1.86 GiB)
[234744.996663] sd 7:0:0:0: [sdb] Write Protect is off
[234744.996669] sd 7:0:0:0: [sdb] Mode Sense: 45 00 00 08
[234744.997518] sd 7:0:0:0: [sdb] No Caching mode page found
[234744.997524] sd 7:0:0:0: [sdb] Assuming drive cache: write
     ➥ through
[234745.009375] sdb: sdb1
[234745.015113] sd 7:0:0:0: [sdb] Attached SCSI removable disk
```

现在 USB 驱动器被识别为/dev/sdb，这样就可以继续使用 dd 命令复制镜像：

```
# dd if=kali-linux-light-2017.1-amd64.iso of=/dev/sdb
2070784+0 records in
2070784+0 records out
1060241408 bytes (1.1 GB, 1011MiB) copied, 334.175s, 3.2MB/s
```

请注意，要确保这些命令执行成功需要 root 权限，而且要确认 USB 驱动器是未使用的，即不存在被挂载的分区。这个命令假设在 ISO 镜像文件所在目录下运行，否则需要提供 ISO 文件的完整路径。

命令中 `if` 的含义是“输入文件（input file）”，`of` 代表“输出文件（output file）”。`dd` 命令从输入文件读取数据，并把它写入输出文件中。它并不会显示任何的进度，所以需要在运行过程中耐心等候（有时候可能要等 1 个小时或更长时间）。观察 U 盘上的写入指示灯是否闪亮（如果有的话），这能帮助你确定命令是否执行完毕。在 OS X/macOS 系统上，也可以在过程中按 Ctrl+T 组合键观察数据复制进度。

在OS X/macOS上创建Kali的启动U盘

OS X / macOS 基于 UNIX，因此在其上创建可启动 Kali Linux USB 驱动器的过程与 Linux 上的过程类似。下载并验证了所选择的 Kali ISO 文件后，使用 `dd` 命令将其复制到 U 盘上。

要识别 U 盘的设备名称，运行 `diskutil list` 命令以列出系统上可用的磁盘。接下来，插入 USB 驱动器并再次运行 `diskutil list` 命令。第二次命令输出应该列出一个额外磁盘。可通过比较两个命令输出来确定 U 盘的设备名称。寻找到这行标识 USB 驱动器的信息，并记下`/dev/diskX`，其中 X 表示磁盘 ID。

应该确保 USB 驱动器并没有被挂载，可通过 unmount 命令来确认（假设`/dev/disk6` 是 U 盘的设备名称）：

```
$ diskutil unmount /dev/disk6
```

现在继续执行 `dd` 命令。这一次，添加一个补充参数表示数据块大小。它定义从输入文件一次读取数据块的大小，然后写入输出文件。

好了，USB 驱动器现已准备就绪，你可以从它启动或使用它来安装 Kali Linux 了。

```
# dd if=kali-linux-light-2017.1-amd64.iso of=/dev/disk6 bs=1M
1011+0 records in
1011+0 records out
1060241408 bytes transferred in 327.061 secs (3242328 bytes/sec)
```

在 OS X/macOS 上从另一个驱动器引导系统

如果想要在一台 OS X/macOS 设备上从另一个驱动器引导系统，则需要在按下电源开关后快速按住 Option 键，然后在弹出的启动菜单中选择想要用来引导系统的驱动器。

可以从苹果公司的知识库中了解更多信息。[1]

1 http://support.apple.com/kb/ht1310

2.2　使用自生模式启动Kali ISO

2.2.1　在一台物理计算机上启动

作为先决条件，你需要准备一个 U 盘（上一节已描述如何制作它）或一个刻录有 Kali Linux ISO 镜像的 DVD-ROM。

BIOS / UEFI 负责早期启动过程，其可通过一个称为 Setup 的 BIOS 内嵌软件进行配置，它允许用户选择优先使用哪个引导设备。根据创建介质，这一步应选择 DVD-ROM 驱动器或 USB 驱动器。

启动 Setup 程序通常要在计算机通电后立刻按下特定键。这个键经常是 `Del` 或 `Esc`，有时是 `F2` 或 `F10`。大多数情况下，当计算机开机后，操作系统加载前，这个选项会在屏幕上短暂闪烁。

BIOS / UEFI 被正确配置为从你的设备启动后，启动 Kali Linux 只是插入 DVD-ROM 或插入 USB 驱动器并打开计算机电源的问题。

禁用安全启动选项　虽然 Kali Linux 镜像可在 UEFI 模式下启动，但它并不支持安全启动（secure boot），你应当在 Setup 程序中关掉这个选项。

2.2.2　在一台虚拟机中启动

虚拟机对于 Kali Linux 用户来说具有很多优势，特别是当你想尝试使用 Kali Linux 但又尚未准备好持久化安装，或者你拥有一台强大的主机并希望同时运行多个操作系统时。这是许多渗透测试人员和安全专业人士的热门选择，他们既需要使用 Kali Linux 中的各种工具，又希望能够完全访问自己的宿主操作系统。这也能够使他们在不需要重新安装操作系统的情况下对工作内容进行存档，或是安全删除虚拟机以及其中可能包含的任何客户数据。

使用虚拟化软件的快照功能还可以试验一些具有潜在危险的操作（如恶意软件分析），通过恢复到先前快照就可以轻松实现。

许多虚拟化工具都可在几乎所有常见的操作系统上使用，这些工具包括 VirtualBox®、VMware Workstation®、Xen、KVM 和 Hyper-V 等。你可以使用最适合你的产品，本书中我们会介绍桌面环境中最常用的两个：在 Windows 10 上运行的 VirtualBox®和 VMware Workstation Pro®。如果没有公司硬性要求或个人偏好，我们建议使用 VirtualBox，因为它是免费的并且运行良好，也（几乎）是开源的，并可用于大多数操作系统。

下面我们假设你已经安装了相应的虚拟化软件并已熟悉其操作。

概述

为了使用虚拟化软件，你应该拥有一个具有相应虚拟化功能且未被 BIOS/UEFI 禁用的 CPU。在 BIOS Setup 屏幕检查是否具有“Intel® Virtualization Technology”或“Intel® VT-d Feature”选项。

同样需要一个 64 位的操作系统，例如基于 Debian 的 Linux 发行版的 amd64 架构、基于 RedHat Linux 发行版的 x86_64 架构，以及 Windows 发行版的 64bit 架构。

如果缺少任何条件，虚拟化工具将无法正常工作，或者仅限于运行 32 位客户机操作系统。

由于虚拟化工具是在主机操作系统及硬件的底层进行挂钩，它们之间往往出现不兼容的情况。所以不要指望这些工具能够同时在一起运行良好。另外，请注意，Windows 专业版可能已经安装并启用了 Hyper-V，这可能会干扰对虚拟化工具的选择。要将其关闭，请从 Windows Settings 中执行“打开或关闭 Windows 功能”。

VirtualBox

初始化安装后，VirtualBox 的主屏幕应如图 2.6 所示。

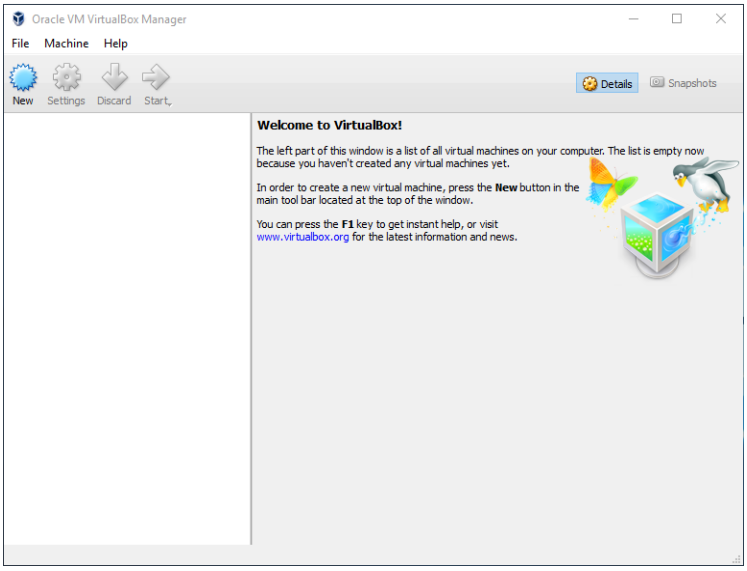


图2.6　VirtualBox的开始画面

单击 New 按钮（见图 2.7）启动向导，该向导将引导你完成输入新虚拟机所有参数所需的多个步骤。

第一步如图 2.7 所示，必须为新虚拟机分配一个名称。我们使用了“Kali Linux”。你必须指明使用何种操作系统。由于 Kali Linux 基于 Debian GNU / Linux，所以选择 Linux 的类型，Debian（32 位）或 Debian（64 位）版本。尽管选择任何其他 Linux 版本都有可能正常运行，但这将帮助区分已安装的其他虚拟机。

第二步，必须确定分配给虚拟机多少内存。虽然推荐给作为服务器的 Debian 虚拟机的内存大小为 768 MB，但用来运行 Kali 桌面系统是绝对不够的。特别是对于 Kali Linux 自生系统，因为自生系统使用内存来存储对文件系统的更改。我们建议将值至少增加到 1500 MB（见图 2.8），并强烈建议分配不少于 2048 MB 的 RAM。

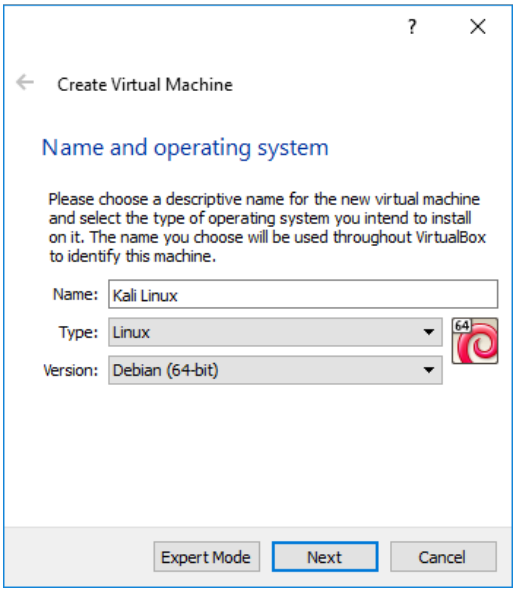


图2.7　名称和操作系统

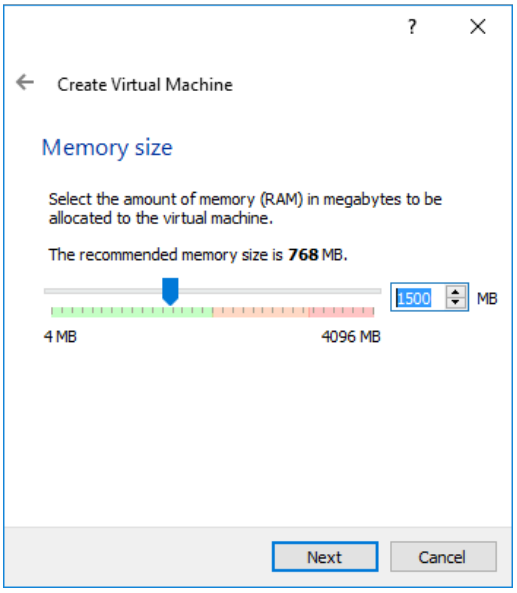


图2.8　内存大小

第三步（如图 2.9 所示），系统将提示为新的虚拟机选择物理或虚拟硬盘。尽管运行 Kali Linux 自生系统并不需要硬盘，但此时还是要为稍后在第 4 章中演示安装程序添加一个硬盘。

虚拟机硬盘内容会作为文件存储在宿主机上。VirtualBox 可使用复合格式存储硬盘内容（如图 2.10 所示）。默认的（VDI）对应 VirtualBox 原生格式；VMDK 是 VMware 使用的格式；QCOW 是 QEMU 使用的格式。这里保持默认值，无须更改。只有把虚拟机从一个虚拟化工具迁移到另一个时，才会要求使用多种文件格式。

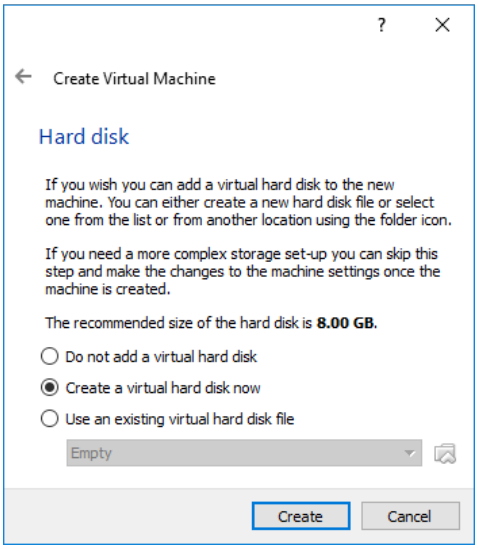


图2.9　硬盘配置

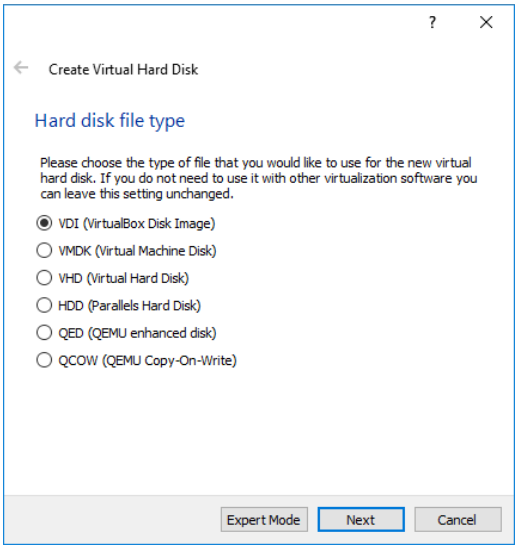


图2.10　硬盘文件类型

图 2.11 中所示的说明文本清楚地描述了动态和固定磁盘分配的优缺点。在这个例子中，笔者接受默认选择（动态分配），因为笔者使用了配备 SSD 磁盘的笔记本电脑，不想浪费空间，也不需要额外性能，它已经足够快了。

如图 2.12 所示，默认硬盘大小为 8 GB，这对于标准安装的 Kali Linux 是不够的，因此要将大小增加到 20 GB。还可以调整磁盘镜像的名称和位置。当硬盘空间不足时，可以方便地将磁盘镜像存储在外部驱动器上。

虚拟机已经创建好了，但由于没有安装操作系统，因此无法真正运行。此时仍然需要调整一些设置，单击 VM Manager 屏幕上的 Settings 按钮，我们来看一些最有用的设置，如图 2.13 所示。

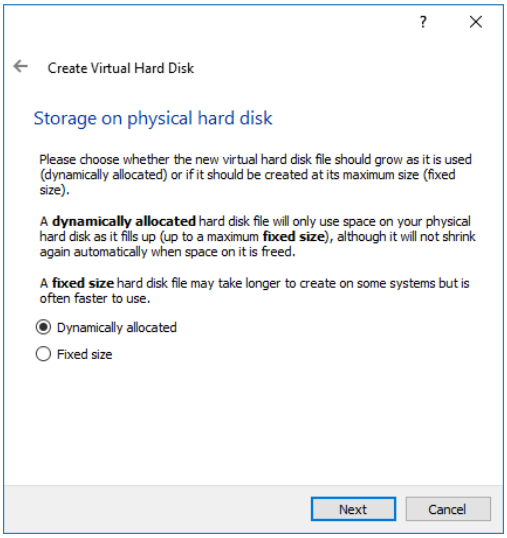


图2.11　在物理磁盘上的存储方式

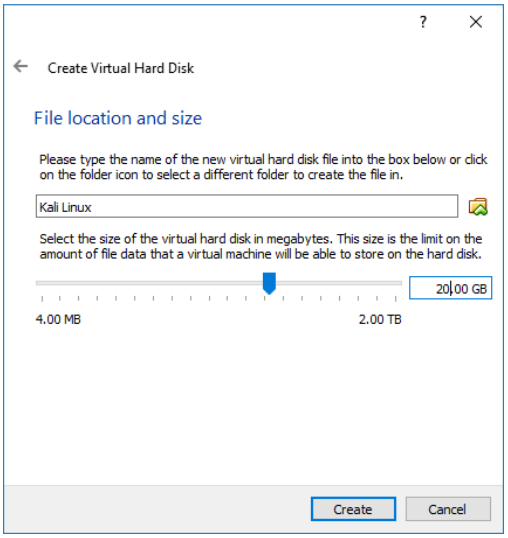


图2.12　文件位置和大小

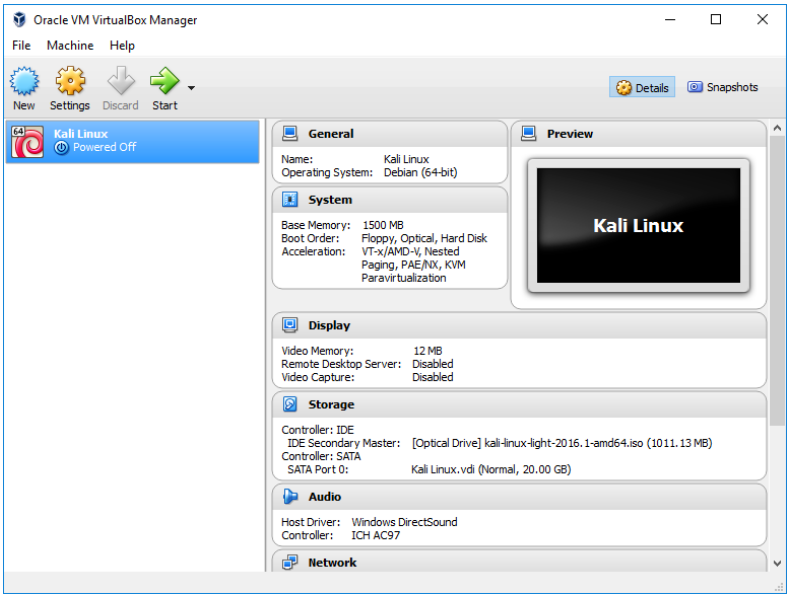


图2.13　新的虚拟机出现在清单中

在 Storage 屏幕（见图 2.14）上，应将 Kali Linux ISO 镜像与虚拟 CD/DVD-ROM 读取器关联。首先，在 Storage Tree 列表中选择 CD-ROM 驱动器，然后单击右侧小 CD-ROM 图标，

以显示上下文菜单。

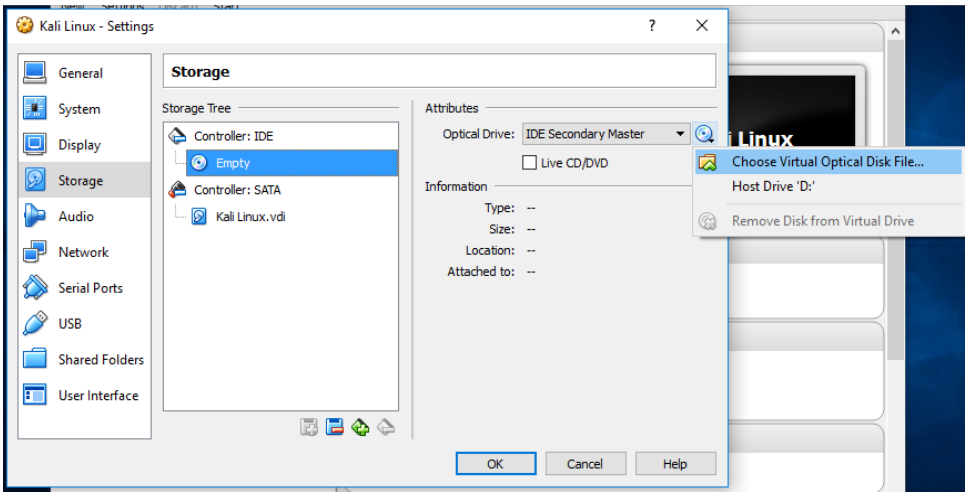


图2.14　存储设置

在 System 屏幕（见图 2.15）上，找到 Motherboard 选项卡。确保引导顺序为：在尝试硬盘启动前首先尝试从任何光学设备启动。在这个选项卡中，也可根据需要改变分配给虚拟机的内存容量。

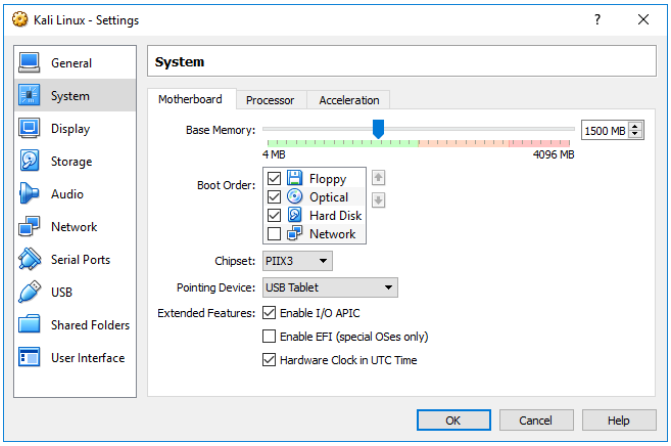


图2.15 系统设置：主板

在位于同一屏幕的 Processor 选项卡上（见图 2.16），可以调整分配给虚拟机的处理器数量。最重要的是，如果使用 32 位镜像，则需要启用 PAE / NX，否则 Kali 镜像将不会启动，因为 Kali for i386（正式命名为“686-pae”）使用的内核变量被编译时需要 CPU 中的 Physical Address Extension (PAE)支持。

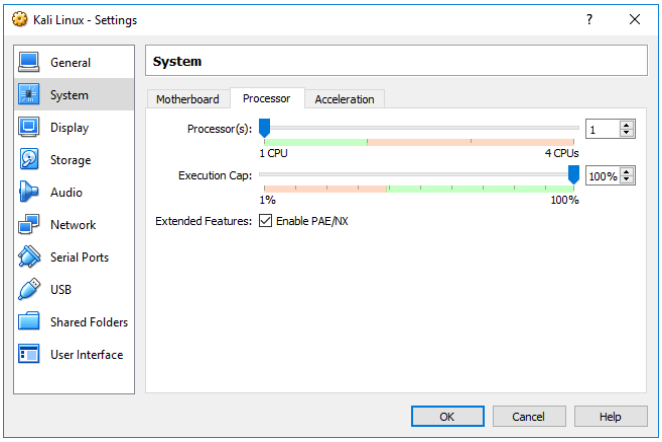


图2.16　系统设置：处理器

我们还可以配置许多其他参数，例如网络设置（定义如何处理网卡上的流量），但上述更改足够启动 Kali Linux 自生系统了。最后，单击 Boot 按钮，虚拟机应该能够正常启动，如图 2.17 所示。如果没有看到这个画面，请仔细检查上述所有设置，然后重试一遍。

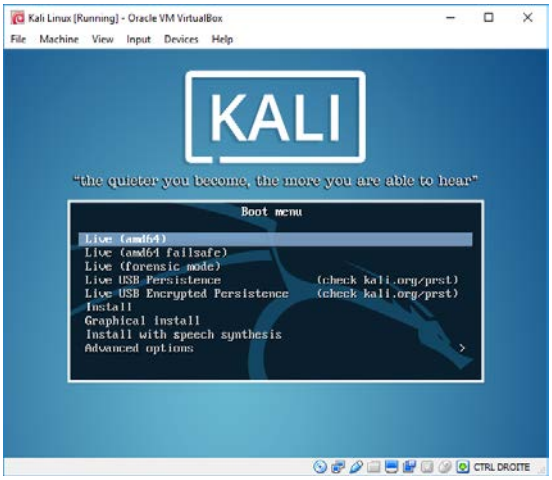


图2.17　Virtual Box中的Kali Linux启动画面

VMware

VMware Workstation Pro 在功能和用户界面方面与 VirtualBox 非常相似，它们都被设计用作桌面环境，但新虚拟机的安装过程有所不同。

如图 2.18 所示，初始屏幕显示一个超大的 Create a New Virtual Machine 按钮，这个按钮可以启动向导，来指导你完成虚拟机的创建。

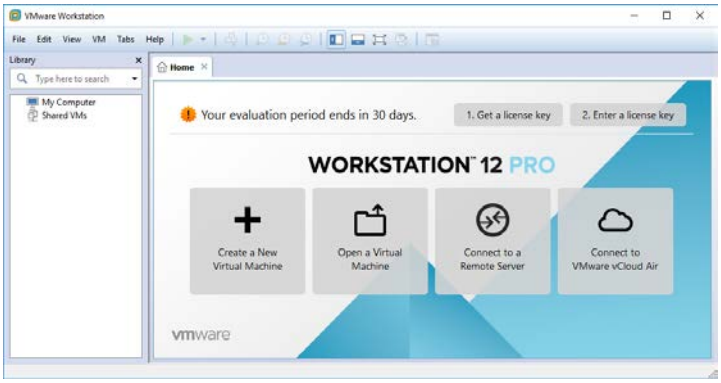


图2.18　VMware启动画面

第一步，必须决定是否在安装过程中呈现高级设置。这个例子没有特殊要求，所以选择典型安装，如图 2.19 所示。

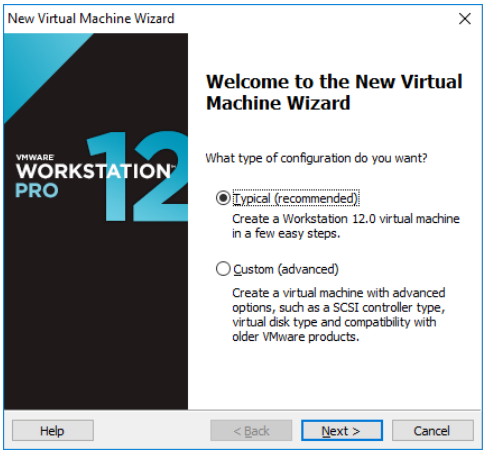


图2.19　新建虚拟机向导

该向导假定要立即安装操作系统，并要求选择包含安装程序的 ISO 镜像（见图 2.20）。选择 Installer disc image file（iso）选项，然后单击 Browse 按钮以选择镜像文件。

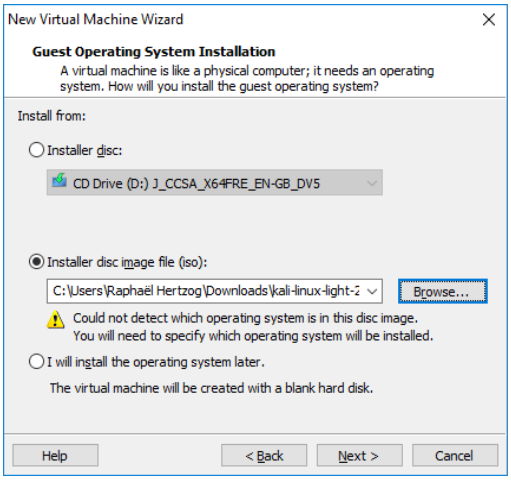


图2.20　安装客户机操作系统

当操作系统（OS）无法从所选 ISO 镜像中检测到客户机操作系统类型时，安装向导会向你询问。应该选择 Linux，版本选择“Debian 8.x”，如图 2.21 所示。

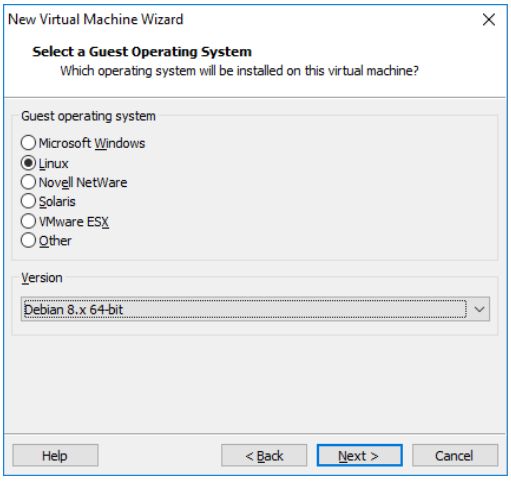


图2.21　选择客户机操作系统

使用“Kali Linux”作为新虚拟机的名称（见图 2.22）。与 VirtualBox 一样，还可以选择将 VM 文件存储在其他备用位置。

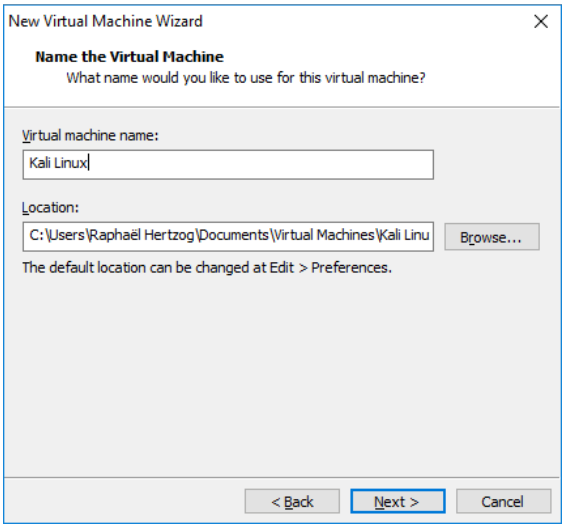


图2.22　命名虚拟机

默认的 20GB 硬盘（见图 2.23）通常是足够的，但可根据需要进行调整。与使用不同大小单个文件的 VirtualBox 相反，VMware 可以将磁盘内容存储在多个文件中。这两种情况的目的都是节省主机的磁盘空间。

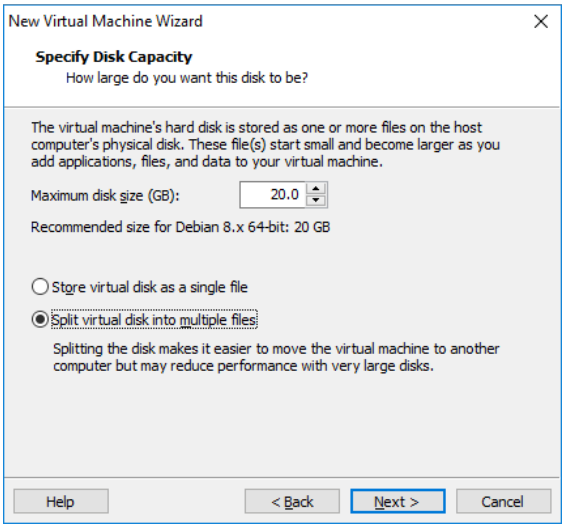


图2.23　指定磁盘容量

VMware Workstation 现在已准备好创建新的虚拟机。它显示所做选择的摘要，以便在创

建机器之前仔细检查所有选项。请注意，向导仅为虚拟机分配了 512 MB 的 RAM，这是不够的，请单击 Customize Hardware 按钮（如图 2.24 所示）并调整 Memory 设置（如图 2.25 所示）。

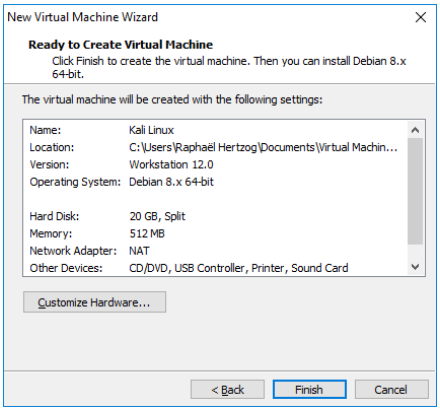


图2.24　已准备好创建虚拟机

单击 Finish 按钮（见图 2.24）。现在可以通过单击 Power on this virtual machine 按钮启动虚拟机，如图 2.26 所示。

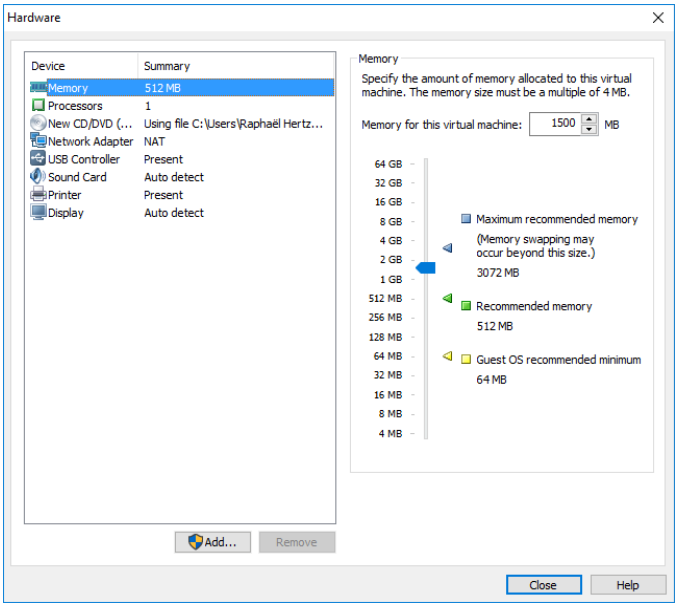


图2.25　配置硬件窗口

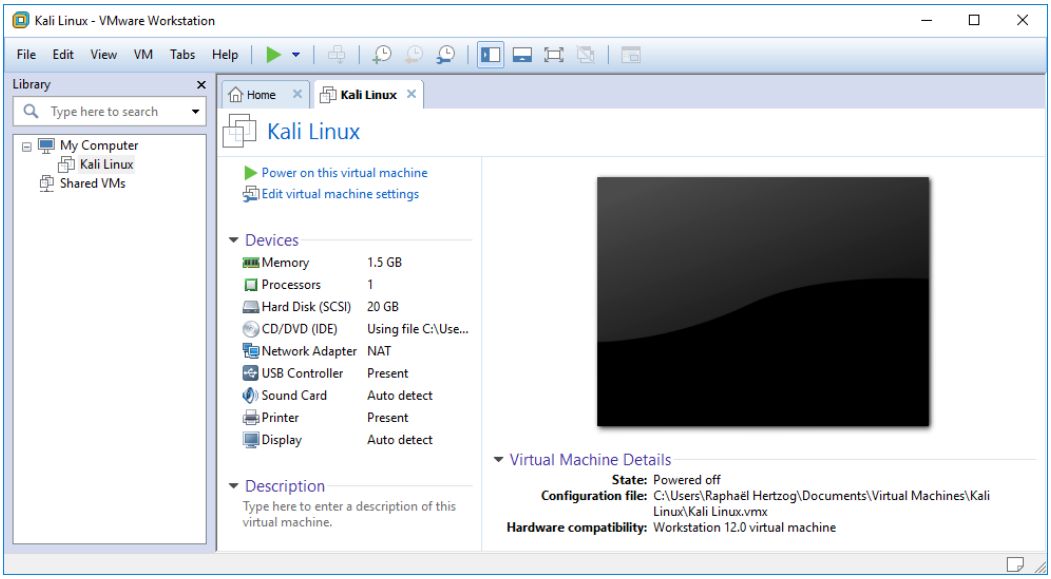


图2.26　Kali Linux虚拟机就绪

2.3　小结

在本章中，我们学习了各种 Kali Linux ISO 镜像以及如何验证并下载它们，并了解了如何在不同操作系统上创建可引导的 USB 磁盘镜像。本章还讨论了如何引导 USB 磁盘镜像，如何在各种硬件平台上配置 BIOS 和启动项，以便使 USB 磁盘镜像启动。

要点提示：

- www.kali.org 是 Kali ISO 的唯一正式下载网站。不要从任何其他网站下载，因为那些下载可能包含恶意软件。
- 总是使用 `sha256sum` 命令来验证下载的 ISO 文件，以确保其完整性。如果不匹配，请再次尝试下载或使用其他来源。
- 如果要在物理机器上引导，则必须将 Kali Linux ISO 镜像写入可引导介质。在 Windows 上使用 `Win32 Disk Imager`，在 Linux 上使用 Disks 实用程序，或在 Mac OS X / macOS 上使用 `dd` 命令。写镜像时要非常小心，如果选择了错误的磁盘，可能会永久丢失计算机上的数据。
- 在 PC 上配置 BIOS / UEFI 设置屏幕，或者按住 OS X / macOS 上的 Option 键，以允许

机器从 USB 驱动器启动。

- 如果想尝试使用 Kali Linux，但又未准备好将其持久性安装，或者你拥有一台强大的主机希望同时运行多个操作系统，那么使用如 VirtualBox 和 VMware Workstation Pro 之类的虚拟机程序，会特别有用。

安装好了 Kali Linux 后，下面可以深入了解 Linux 中的一些高级功能了。如果你已经是 Linux 的熟练用户，则可以考虑跳过下一章。

练习题

练习1——安装、下载、验证和烧录Kali

1. 安装一个虚拟机软件，如 VMware Fusion（OSX）、VirtualBox 等。
2. 下载一个 Kali Linux 的虚拟机预装版本（如 64 位版）。
3. 启动 Kali 虚拟机。
4. 以下步骤在虚拟机中完成：登录到虚拟机（使用用户名 root，密码 toor），从 https://www.kali.org/downloads/下载“Kali 64 bit”ISO 镜像文件并保存到虚拟机上。
5. 下载并导入 Kali 的公钥。
6. 从 Kali ISO 中导出指纹并获取 SHA256SUM 以及相关的签名文件。
7. 确认下载的 sha256 校验值同 SHA256SUM 文件一致。
8. 从下载的镜像创建一个能够启动的 USB 驱动器。

练习2——启动Kali

1. 使用上一个练习中创建的 USB 驱动器启动 Kali，并选择 Live 自生模式。
2. 在/root 目录下面创建一个 6GB 大小的文件。
3. 发生了什么？原因是什么？
4. 重启系统，确认对系统的改变是非持久性的。

练习3——修改启动参数

1. 我们已经从一个预制的虚拟机和一个 USB 驱动器中启动了 Kali。现在我们试一下另一

种启动方法：从 Kali ISO 镜像文件启动虚拟机。确认网络已经设置为 NAT 模式。

2. 修改 Live 自生启动参数，在内核行添加“quiet”参数以显示更少的启动日志。

3. 在启动时观察屏幕以确认这个参数已生效并减少了启动日志。

4. 分别用 Live 和 Forensics 参数启动系统，并指出它们的区别。

思考题

1．你能想到哪些场景适合使用自生模式启动 Kali 吗？哪些场景不适合？

2．深度问题：你可以简单地使用 dd 命令将 ISO 写入 USB 磁盘，并使用它引导系统，对此你有没有感到惊讶呢？简述这是基于什么原理做到的。

第3章　Linux基础

内容

- 什么是 Linux
- 它能干什么
- 命令行
- 文件系统
- 有用的命令
- 小结

在你掌握 Kali Linux 之前，必须习惯使用通用 Linux 系统。因为大部分网络、电子邮件和其他互联网服务都在 Linux 系统上运行，掌握 Linux 知识将非常有用。

本章尽量涵盖 Linux 基础知识，但假设你已经对计算机系统有了一般性的了解，包括 CPU、RAM、主板和硬盘等组件，以及设备控制器及其相关连接器等。

3.1　什么是Linux，它能做什么

“Linux”这个术语经常用于指代整个操作系统，但实际上，Linux 是操作系统内核，它由引导加载程序（boot loader）启动，而该引导加载程序本身由 BIOS / UEFI 启动。内核承担与管弦乐队指挥相似的角色——确保硬件和软件的协调。内核包括管理硬件、进程、用户、权限和文件系统。内核为系统所有其他程序提供了一个共同的基础平台，通常在 `ring 0`（也称为内核空间）里运行。

用户空间（User Space）	我们使用名词“用户空间”统称所有在内核外发生的一切。 在用户空间运行的程序中，有很多核心应用来自于GNU project[1]，它们中

1 http://www.gnu.org

的大多数需要在命令行中运行。可以把它们用在脚本中以完成各种自动化任务。可参阅 3.4 节了解到更多信息。

下面我们来快速了解一下 Linux 内核处理的各种任务。

3.1.1 驱动硬件设备

内核的任务首先是控制计算机硬件组件。当计算机开机、插入或取出设备（例如 USB 设备）时，内核会检测到并配置硬件。内核还通过简化编程接口，使它们可以被更高级别软件调用，这样应用程序就无须解决诸如“哪个可选的板卡插入了主板上哪条扩展槽”等细节问题而方便地使用设备。编程接口还提供一个抽象层。举个例子，视频会议软件希望可以使用任何厂家和型号的网络摄像头，那么该软件可以使用 Video for Linux（V4L）接口，内核将会把接口函数转换为使用特定网络摄像机所需的实际硬件命令。

内核通过`/proc/`和`/sys/`虚拟文件系统导出检测到的硬件数据。应用程序通常通过在`/dev/`中创建的文件访问设备。特定文件表示磁盘驱动器（例如`/dev/sda`）、分区（`/dev/sda1`）、鼠标（`/dev/input/mouse0`）、键盘（`/dev/input/event0`）、声卡（`/dev/snd/*`）、串行端口（`/dev/ttyS*`）和其他组件。

有两种类型的设备文件：块和字符。前者具有数据块特征，它大小有限，你可访问块中任何位置的字节。后者表现得像一个字符流，你可以读取和写入字符，但不能寻找给定位置并更改任意字节。要查找给定设备文件的类型，检查 `ls -l` 命令输出中的第一个字母。它可以是 b，表示块设备，也可以是 c，表示字符设备：

```
$ ls -l /dev/sda /dev/ttyS0
brw-rw---- 1 root disk 8, 0 Mar 21 08:44 /dev/sda
crw-rw---- 1 root dialout 4, 64 Mar 30 08:59 /dev/ttyS0
```

正如你所预期的那样，磁盘驱动器和分区使用块设备，而鼠标、键盘和串行端口使用字符设备。在这两种情况下，编程接口都包括可以通过 ioctl 系统指令进行引用的设备相关命令。

3.1.2 统一文件系统

文件系统是内核的重要组成。类 UNIX 系统将所有文件存储合并到一个统一的层级结构中，这个结构允许用户和应用程序通过层次结构中的位置来访问数据。

这个分层树的起始点称为根，由“/”字符表示。此目录可以包含多个命名的子目录。例

如，/的 home 子目录为/home/。这个子目录又可以包含其他子目录，依此类推。每个目录都包含存储数据的文件。因此，/home/buxy/Desktop/hello.txt 指的是一个存储在根目录的 home 子目录 buxy 子目录 Desktop 目录中的文件。内核在这套命名系统和磁盘存储位置之间进行转换。

与其他系统不同，Linux 只有一个这样的层级结构，同时它可以集成多个物理磁盘数据。这些磁盘中的某一个成为根，其他磁盘成为层次结构中的目录（相应的 Linux 命令为 mount），之后在挂载点（mount point）下便可以访问这些磁盘了。这样就允许将用户的主目录（通常是/home/）存储在单独硬盘上，该硬盘将包含 buxy 目录（以及其他用户的主目录）。一旦将磁盘挂载在/home/下，这些目录就可以在其常用位置和路径（如/home/buxy/Desktop/hello.txt）中被访问。

与各种磁盘上物理存储数据方式相对应，有许多种文件系统格式。最著名的是 ext2、ext3 和 ext4，还有其他格式。例如，VFAT 是 DOS 和 Windows 操作系统曾经使用的文件系统。Linux 支持 VFAT，允许像在 Windows 下一样在 Kali 中访问这种磁盘格式。但无论如何，在挂载磁盘之前必须在磁盘上准备一个文件系统，此操作称为格式化。

诸如 mkfs.ext3（其中 mkfs 表示 MaKe FileSystem）之类的命令用于处理格式化操作。作为参数，该命令需要被格式化分区的设备文件标识（例如，/dev/sda1，第一个驱动器上的第一个分区）。此操作是破坏性的，只应在设备初始化时运行一次，除非想擦除文件系统并清空所有数据。

诸如 NFS 之类的网络文件系统不在本地磁盘存储数据，而是将数据通过网络传输到能够存储和检索数据的服务器上。得益于文件系统抽象技术，你无须担心磁盘如何连接，因为其访问方式同文件系统常见的层次结构是一致的。

3.1.3　管理进程

所谓进程是指一个程序的运行实例，它需要存储器来存储程序本身及其操作数据。内核负责创建和跟踪进程。当程序运行时，内核首先准备好一些内存，将可执行代码从文件系统加载到内存里，然后开始运行代码。内核保存此进程的运行信息，其中最常见的是称为进程标识符（PID）的数字。

像大多数现代操作系统一样，具有类 UNIX 内核的系统（包括 Linux）能够进行多任务处理。换句话说，它们允许系统同时运行许多进程。实际上同一时刻只有一个进程在运行，但内核将 CPU 时间划分成小片，依次运行每个进程。由于这些时间片非常短（在毫秒范围内），所以尽管它们仅在其运行时间周期内处于活动状态而在其余时间处于空闲状态，程序

进程仍然看起来像是在同时运行。内核的工作是调整其调度机制以保持性能，同时最大化全局系统性能。如果分配的时间片太长，应用程序可能不会根据需要及时响应。如果太短，那么系统就会因为频繁切换任务而浪费 CPU 时间。这些决策都可以通过调整进程优先级进行改进，高优先级进程将比低优先级进程运行时间更长，获得时间片更频繁。

多处理器系统　上面所描述的同一时间只能运行一个进程的限制并不总是成立。准确的说法是，每一个处理器核心同一时间只能运行一个进程。多处理器、多核心或者超线程系统允许多个进程并行运行。时间分片机制仍然在使用，以便处理进程数超过处理器核心数量的情况。实际上处理器核心数量相对于进程数总是不够，即使是一个空闲的操作系统也会有几十个进程同时在运行。

内核允许相同程序的几个实例独立运行，但每个实例仅允许访问其自己的时间片和内存。它们的数据相互隔离。

3.1.4　权限管理

类 UNIX 系统支持多个用户和组，以方便控制访问权限。大多数情况下，一个进程的权限由启动它的用户来决定。该进程只能执行所有者所能执行的操作。例如，是否有权限打开某个文件需要内核根据进程标识去检查它的访问权限（有关此具体示例的详细信息，请参见 3.4.4 节的内容）。

3.2　命令行

“命令行”指一个基于文本的界面，这个界面允许输入命令执行它们并查看结果。可以运行一个命令行终端（图形桌面中的文本屏幕或非图形界面的文本控制台本身）或命令解释器（shell）。

3.2.1　如何获得一个命令行

当系统正常工作时，访问命令行最简单的方法是在图形桌面会话中打开一个终端。

例如，在默认的 Kali Linux 系统上，可以从常用应用程序列表中启动 GNOME 终端。还可以在“活动”屏幕（当你将鼠标移动到左上角时被激活）中键入“terminal”，然后单击随

后出现的应用程序图标（见图 3.1）。

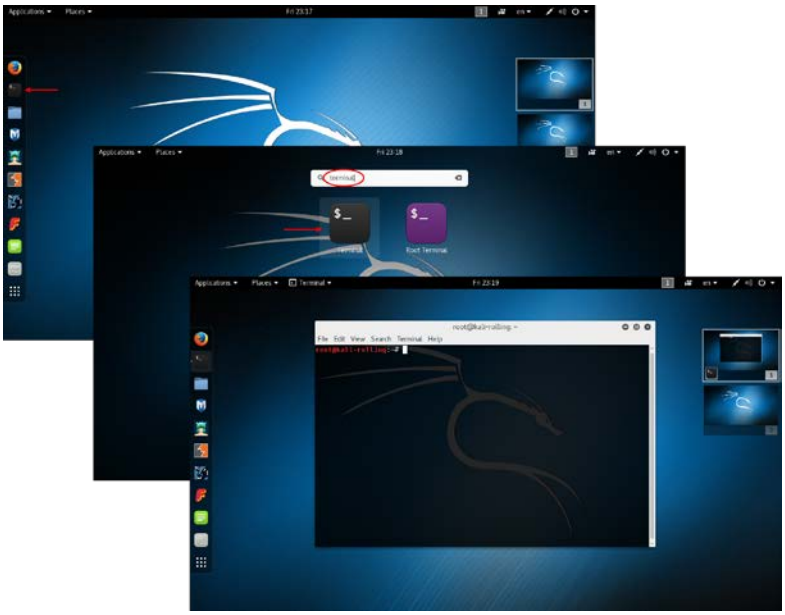


图3.1　启动GNOME终端

如果图形界面被损坏无法使用，仍可在虚拟控制台上获得命令行（最多可以通过 CTRL + ALT + F1~CTRL + ALT + F6 这 6 个组合键开启 6 个终端——如果已处于文本模式并在 Xorg 或 Wayland 图形界面之外，可以不用按下 CTRL 键）。此时将看到一个登录屏幕，输入登录名和密码后才能通过其 shell 访问命令行：

```
Kali GNU/Linux Rolling kali-rolling tty3
kali-rolling login: root
Password:
Last login: Fir Mar 25 12:30:05 EDT 2016 from 192.168.122.1 on pts/2
Linux kali-rolling 4.4.0-kali1-amd4 #1 SMP Debian 4.4.6-1kali1 (2016-03-18)
     ➥ x86_64

The programs included with the Kali GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
root@kali-rolling:~#
```

处理输入和执行命令的程序称为 shell（或命令行解释器）。在 Kali Linux 中提供的默认 shell 是 Bash（它代表 Bourne Again SHell）。后缀的“$”或“#”字符表示 shell 正在等待输入。它还指明了 Bash 将你视为普通用户（用“$”表示）或超级用户（使用“#”表示）。

3.2.2 命令行基础：浏览目录树以及管理文件

本章仅简要地概述相关命令，所有命令都有很多选项，因此请参考命令手册页面中丰富的文档，以了解更多使用方法。在渗透测试中，通常会在一次成功漏洞利用后得到系统的 shell 入口，而并非图形用户界面。因此熟练使用命令行对于你成为成功的安全专家至关重要。

会话打开后，`pwd` 命令（代表 print working directory，即打印工作目录）能够显示当前你在文件系统中的位置。更改当前目录使用 `cd` 命令（代表 change directory，即更换目录）。如果 `cd` 命令未指定目标目录，你将被带到主目录。当使用 `cd-`命令时，将返回到前一个工作目录（最后一次 `cd` 命令调用之前使用的目录）。父目录始终被记为..（两个点），而当前目录被记为.（一个点）。`ls` 命令能够列出目录内容，如果不提供任何参数，`ls` 命令将对当前目录进行操作。

```
$ pwd
/home/buxy
$ cd Desktop
$ pwd
/home/buxy/Desktop
$ cd .
$ pwd
/home/buxy/Desktop
$ cd ..
$ pwd
/home/buxy
$ ls
Desktop Downloads Pictures Templates
Documents  Music  Public   Videos
```

可以使用 `mkdir` directory 来创建新目录，并使用 `rmdir` directory 删除现有的（空）目录。`mv` 命令允许移动和重命名文件及目录; `rm` 命令实现文件的删除，`cp` source-file target- file 命令能够完成文件复制。

```
$ mkdir test
$ ls
Desktop Downloads Pictures Templates Videos Documents Music Public test
```

```
$ mv test new
$ ls
Desktop Downloads new Public Videos Documents Music Pictures Templates
$ rmdir new
$ ls
Desktop Downloads Pictures Templates Videos Documents  Music     Public
```

shell 通过运行在 PATH 环境变量目录清单中的第一个位置，找到应用程序执行每个命令。这些程序通常在/bin、/sbin、/usr/bin 或/usr/sbin 中。例如，可以在/bin/ls 中找到 ls 命令；which 命令能够报告给定可执行文件的位置。有些命令由 shell 直接处理，在这种情况下，其被称为 shell 内置命令（如 cd 和 pwd 命令）；type 命令可以查询每个命令类型。

```
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
$ which ls
/bin/ls
$ type rm
rm is /bin/rm
$ type cd
cd is a shell builtin
```

注意 echo 命令的用法，它仅仅在终端上显示一个字符串。在上面的例子中，它用来打印环境变量的内容，因为在执行命令行之前，shell 会自动用变量值替换变量名称（$PATH）。

环境变量　环境变量允许 shell 或其他程序存储全局设置。例如，每个进程有它自己的一套环境变量（它们是上下文相关的）。Shell 程序，例如 login shell，能够声明变量，这些变量能够传递到被它们启动的其他程序中去（可继承）。这些变量可以被定义到系统层面（在/etc/profile 中定义），或用户层面（在~/.profile 中定义），但是那些并不专门用于命令行的变量最好放在/etc/environment 中，在可插拔认证模块（PAM）的帮助下，这些变量能够被注入所有用户会话中，即使没有运行 shell。

3.3　文件系统

3.3.1　文件系统层次标准

与其他 Linux 发行版一样，Kali Linux 的组成也同文件系统分层标准（FHS）保持一致，这样熟悉其他 Linux 发行版的用户也能够轻松使用 Kali。FHS 定义了每个目录的用途。几个

顶层目录设置描述如下。

- /bin/：基本应用程序
- /boot/：Kali Linux 内核及其早期引导过程所需的其他文件
- /dev/：设备文件
- /etc/：配置文件
- /home/：用户个人文件
- /lib/ ：基本库文件
- /media/ *：可移动设备装载点（如 CD-ROM、USB 存储设备等）
- /mnt/：临时挂载点
- /opt/：第三方提供的额外应用程序
- /root/：管理员（root）个人文件
- /run/：重启后就不存在的临时运行时数据（目前还不是 FHS 标准）
- /sbin/：系统程序
- /srv/：由本系统托管服务所使用的数据
- /tmp/：临时文件（此目录通常在启动时被清空）
- /usr/：应用程序（根据与根目录相同的逻辑，该目录被进一步细分为 bin、sbin 和 lib）。此外，/usr/share/包含与架构无关的数据。/usr/local/目录用于管理员手动安装应用程序，而不会覆盖包管理系统（dpkg）所自动处理的文件。
- /var/：由守护进程处理的变量数据，包括日志文件、队列、任务清单以及缓存等。
- /proc/和/sys/：同特定 Linux 内核相关（并非 FHS 标准的一部分）。内核用这两个目录把数据导出到用户空间。

3.3.2 用户的主目录

用户主目录内容要怎样安排并没有标准，但仍然有一些值得注意的约定。一个是用户主目录通常由波浪号（“~”）引用。由于命令解释器会自动将波浪号替换为当前用户目录（存储在环境变量中，通常为/home/user/），所以知道这一点是有用的。

传统上，应用程序配置文件通常直接存储在主目录下，但文件名通常以点（.）开头（例如，mutt 电子邮件客户端将其配置存储在~/.muttrc 中）。请注意，以点开头的文件名默认被隐藏；只有将图形文件管理器配置为显示隐藏文件，或者为 ls 命令添加-a 选项，才能够看到这些文件。

一些程序会使用配置在相同目录下的多个配置文件（例如~/.ssh/）。一些应用程序

（如 Firefox Web 浏览器）还会使用这些目录作为下载数据的缓存。这意味着这些目录可能会消耗大量磁盘空间。

这些直接存储在主目录中的配置文件（通常统称为“点文件”）长期以来一直在不断增加，使得主目录十分混乱。所幸在 FreeDesktop.org 的努力与支持下出台了 XDG 基本目录规范（XDG Base Directory Specification），这是一个旨在清理这些文件和目录的公共约定。该规范规定，配置文件应存储在~/.config 下，缓存文件应存储在`~/.cache` 下，应用程序数据文件应存储在`~/.local`（或其子目录）下。这个惯例正在慢慢形成。

图形界面桌面上通常会显示`~/Desktop/`目录内容的快捷方式（在其他语言中可能会被翻译成相应词汇，如在中文系统中该目录为/桌面/）。

最后，电子邮件系统有时会将收到的电子邮件存储到`~/Mail/`目录中。

3.4　有用的命令

3.4.1　显示和修改文本文件

`cat` *file* 命令（用于将文件输出到标准输出设备）读取文件并将其内容显示在终端上。如果文件太大而无法放在屏幕上，可以使用 `less`（或 `more`）分页器逐页显示。

`editor` 命令能够启动文本编辑器（例如 Vi 或 Nano）并允许创建、修改和读取文本文件。最简单的文件创建方式是通过重定向语句，例如 `command>file` 会创建一个名为 file 的文件，其中包含了 command 命令输出。`command>>file` 与之类似，除了它将 command 输出附加到文件，而不是覆盖它以外。

```
$ echo "Kali rules!" > kali-rules.txt
$ cat kali-rules.txt
Kali rules!
$ echo "Kali is the best!" >> kali-rules.txt
$ cat kali-rules.txt
Kali rules!
Kali is the best!
```

3.4.2　搜索文件和文件内容

`find` directory criteria 命令根据多个条件，在 directory 之下的层次结构中搜索文件。最常

用条件是按名称搜索文件。也可以在文件名搜索中使用常用通配符，如“*”。

```
$ find /etc -name hosts
/etc/hosts
/etc/avahi/hosts
$ find /etc -name "hosts*"
/etc/hosts
/etc/hosts.allow
/etc/hosts.deny
/etc/avahi/hosts
```

grep expression files 命令搜索文件内容并提取与正则表达式匹配的行。添加-r 选项可以对目录包含的所有文件进行递归搜索。这样在只知道一部分文件内容时也可以查找文件。

3.4.3 进程管理

ps aux 命令列出当前运行的程序，并通过显示其 PID 来帮助识别它们。一旦知道程序的 PID，kill -signal pid 命令就允许发送一个终止进程的信号（如果你拥有该进程的权限）。有好几个信号可以使用，最常用的是 TERM（优先终止请求）和 KILL（强制杀死）。

如果命令后跟“＆”符号，则命令解释器也可以在后台运行程序。通过使用＆符号，即使程序仍在运行（从当前视图中隐藏成为后台进程），也可以立即恢复对 shell 的控制。jobs 命令列出在后台运行的进程; 运行 fg % jod-number（意为 foreground）将作业还原到前台。当一个命令在前台运行时（通过正常的 shell 命令启动，或者通过 fg 命令还原到前台），使用 Ctrl +Z 组合键会暂停进程并恢复对命令行的控制。然后可以使用 bg %job-number（意为 background）在后台重新启动该进程。

3.4.4 权限管理

Linux 是一个多用户系统，因此有必要提供权限控制系统，来管理文件和目录上的授权操作集，这同时包括了所有系统资源和设备（在 UNIX 系统上，任何设备都由文件或目录表示）。这个原则对于所有类 UNIX 系统都适用。

每个文件或目录对三类用户具有特定权限：

- 所有者（以 u 表示，意为 user）
- 所有者组（以 g 表示，意为 group），代表所有组成员
- 其他（以 o 表示，意为 other）

三种组合权限：

- 读取（以 r 表示，意为 read）
- 写（或修改，以 w 表示，意为 write）
- 执行（以 x 表示，意为 eXecute）

从文件角度来看，这些权限很容易理解。读取权限允许读取内容（包括复制），写入权限允许更改，执行权限允许运行（只有在对象是应用程序时才有效）。

setuid 与 setgid

可执行文件还有两种特殊权限：`setuid` 与 `setgid`（以字母 `s` 表示）。这两个权限允许任何用户使用程序的所有者或所有组的权限去运行它。这个机制能够赋予你通常不具备的权限。

既然一个 setuid root 程序会被以超级用户的身份执行，那么它的安全性和可靠性是必须得到保证的。图谋不轨的用户可以利用 setuid root 程序去调用一个他们自己的命令，从而模仿 root 用户并获取系统的全部权限。渗透测试工程师们一旦获取了系统的访问权，通常会在系统中寻找这类文件，作为提升权限的一种途径。

目录处理方式与文件不同。读取权限可以查看其内容（文件和目录）列表；写入权限允许创建或删除文件；执行权限允许跨目录访问其内容（例如，使用 `cd` 命令）。在能够跨目录访问但没有读取权限的情况下，用户能够访问其中已知名称的条目，但如果不知道确切名称就无法找到它们。

安全

setgid 目录与 sticky bit

setgid 权限也能应用在目录上。在这种目录中，所有新创建的项目都会自动地被设置为其父目录的权限，而不是像通常那样继承创建者的权限。正因如此，当你和几位同属某用户组的多个用户共同在同一个文件树中工作时，你并不需要切换你的主用户组（使用 `newgrp` 命令）。

黏滞位（sticky bit，以字母 t 表示）是一种仅仅用于目录的权限。对于那些所有人都具有写入权限的临时目录（例如`/tmp/`），sticky 权限非常有用。它限制文件被删除，只有文件的所有者或父目录的所有者才能够删除文件。如果不设置这个权限，任何人都以删除/tmp/目录中其他用户的文件。

有三个命令用来控制与文件相关的权限：

- `chown` user file 命令更改文件所有者。

提示	你经常会希望改变文件所有者组的同时也改变其所有者，chown 命令对这种
更改用户和组	操作有一个特别的语法：chown user:group file

- chgrp group file 命令更改所有者组。
- chmod rights file 命令更改文件权限。

有两种表示权限的方式。其中，符号表示可能是最容易理解和记忆的。这种方式用到上面提到的字母符号（u/g/o），可以通过等于号（=）明确指定权限，或是通过加号（+）或减号（–）增加或移除权限。因此，这样一个表达式：u=rwx,g+rw,o-r，意为赋给所有者读取、写入和执行权限，赋给所有者组增加、读取和写入权限，同时移除其他用户读取权限。命令没有涉及的权限不作更改。表示 all 的字母 a，指全部三类用户（所有者、组、其他人），因此 a=rx 将授予全部三个类别用户相同权限（读取和执行，但不能写入）。

可以用（八进制）数字表示权限：4 表示读取，2 表示写入，1 表示执行。我们将每个权限组合与三位数字和进行关联，并按照顺序（所有者、组、其他人）给每个类别的用户分配一个值。

例如，chmod 754 *file* 命令将设置以下权限：所有者读取、写入和执行（因为 7 = 4 + 2 + 1）；组读取和执行（因为 5 = 4 + 1）；其他用户则只读。0 意味着没有任何权限。因此，chmod 600 *file* 指为所有者提供读写权限，但不给其他人提供任何权限。对可执行文件和目录分配的权限最常见的是 755，对数据文件最常见的是 644。

要表示特殊权限，可以根据相同原则为该号码加上前缀变成四位，其中 setuid、setgid 和 sticky 分别用 4、2 和 1 表示。命令 chmod 4754 会在先前描述权限基础上再添加 setuid 权限。

请注意，使用八进制符号仅允许在文件中一次性设置所有权限；不能使用它来添加新权限，例如为用户组添加一个写入权限，因为没有数字能够表达现有权限和待增加或移除权限的关系。

八进制表达方式也可以和用于限制新建文件权限的 umask 命令一起使用。可以理解为，当应用程序创建一个文件时，会将其默认权限设置为 777 减去 umask 定义的权限。在 shell 中输入 umask，你将看到一个诸如 0022 的掩码。这就是新文件会被系统去除权限的八进制表达（0022 表示组成员和其他用户将都不具有写入的权限）。

可以使用 umask 命令设置新八进制掩码。在 shell 初始化文件（例如 ~/.bash_profile）中使用它，将有效地更改工作会话的默认掩码。

技巧 **递归操作**

有时候我们必须改变整个文件树的权限。所有上述命令都带一个`-R`选项，其能够让我们把更改递归地应用到子目录中。

由于目录和文件权限含义的区别，递归操作有时会出现一些歧义。这也是为什么“X”操作符被引入。它表示可执行权限仅仅应用在目录中，而不应用在不具备执行权限的文件上。这样，`chmod -R a+X` directory 将会为全部用户类型、全部子目录添加执行权限，但只会对那些已经在某类用户中具有执行权限的文件添加执行权限。

3.4.5　获取系统信息和日志

`free` 命令显示内存信息；disk free（`df`）报告文件系统中每个磁盘的可用空间。它的`-h`选项（表示人类可读）将尺寸转换为更易于理解的单位，通常是兆字节（mebibytes，简称 MB，即 2^{20} 个字节），或吉字节（gibibytes，简称 GB，即 2^{30} 个字节）。类似的，`free` 命令支持`-m`和`-g`选项，表示分别以 MB 或 GB 为单位显示其数据。

```
$ free
              total        used        free      shared  buff/cache   available
Mem:        2052944      661232      621208       10520      770504     1359916
Swap:             0           0           0
$ df
Filesystem     1K-blocks      Used Available Use% Mounted on
udev             1014584         0   1014584   0% /dev
tmpfs             205296      8940    196356   5% /run
/dev/vda1       30830588  11168116  18073328  39% /
tmpfs            1026472       456   1026016   1% /dev/shm
tmpfs               5120         0      5120   0% /run/lock
tmpfs            1026472         0   1026472   0% /sys/fs/cgroup
tmpfs             205296        36    205260   1% /run/user/132
tmpfs             205296        24    205272   1% /run/user/0
```

`id` 命令显示运行会话的用户标识及其所属组的列表。由于对某些文件或设备的访问可能仅限于组成员，因此检查用户是哪些组的成员通常是必要的。

```
$ id
uid=1000(buxy) gid=1000(buxy) groups=1000(buxy),27(sudo)
```

`uname -a` 命令返回一行记录，描述内核名称（Linux）、主机名、内核发行版、内核版本、机器类型（诸如 x86_64 的架构字符串）以及操作系统名称。这个命令的输出通常应该包

含在应用程序的错误报告中，因为它清楚地定义了正在使用的内核和正在运行的硬件平台。

```
$ uname -a
Linux kali 4.9.0-kali3-amd64 #1 SMP Debian 4.9.18-1kali1 (2017-04-04) x86_64
     ➥ GNU/Linux
```

所有这些命令都能够提供运行时信息，但经常还需要通过分析日志来了解计算机上发生的情况。特别是，每当发生重要的事情时（例如插入新 USB 设备，硬盘操作发生故障或启动时进行初始硬件检测），内核就会发布存储在环形缓冲区中的消息。可以使用 `dmesg` 命令获取内核日志。

systemd 的 journal 也存储了多个日志（stdout/stderr 输出的守护进程、syslog 消息及内核日志），可以使用 `journalctl` 命令方便地查询它们。如果不加任何选项，这个命令仅仅按时间顺序转储所有可用日志。使用`-r` 选项，它将反转顺序，先显示较新消息。使用`-f` 选项，它将不断打印新日志条目，并将它们附加到其数据库。`-u` 选项可以将消息限制为由特定 systemd 单元发布的消息（例如 `journalctl -u ssh.service`）。

3.4.6 发现硬件

内核通过/proc/和/sys/虚拟文件系统，导出许多被检测到的硬件细节。一些工具能够汇总分析这些细节。其中，`lspci`（在 pciutils 包中）能够列出 PCI 设备，`lsusb`（在 usbutils 包中）能够列出 USB 设备，`lspcmcia`（在 pcmciautils 包中）能够列出 PCMCIA 卡。这些工具对于识别设备确切型号非常有用。有了这些型号，我们能够在互联网上进行更精确的搜索，从而找到更多文档。请注意，pciutils 和 usbutils 包已经安装在 Kali 基础系统上，但是 pcmciautils 必须使用 `apt install pcmciautils` 安装。稍后我们将讨论有关软件包安装和管理的更多信息。

示例 3.1　`lspci` 和 `lsusb` 提供的信息

```
$ lspci
[...]
00:02.1 Display controller: Intel Corporation Mobile 915GM/GMS/910GML
     ➥ Express Graphics Controller (rev 03)
00:1c.0 PCI bridge: Intel Corporation 82801FB/FBM/FR/FW/FRW (ICH6 Family)
     ➥ PCI Express Port 1 (rev 03)
00:1d.0 USB Controller: Intel Corporation 82801FB/FBM/FR/FW/FRW (ICH6 Family)
     ➥ USB UHCI #1 (rev 03)
[...]
01:00.0 Ethernet controller: Broadcom Corporation NetXtreme BCM5751 Gigabit
```

```
     ➥ Ethernet PCI Express (rev 01)
02:03.0 Network controller: Intel Corporation PRO/Wireless 2200BG Network
     ➥ Connection (rev 05)
$ lsusb
Bus 005 Device 004: ID 413c:a005 Dell Computer Corp.
Bus 005 Device 008: ID 413c:9001 Dell Computer Corp.
Bus 005 Device 007: ID 045e:00dd Microsoft Corp.
Bus 005 Device 006: ID 046d:c03d Logitech, Inc.
[...]
Bus 002 Device 004: ID 413c:8103 Dell Computer Corp. Wireless 350 Bluetooth
```

这些程序有一个`-v` 选项，其能够列出更详细的信息（但通常没有必要），最后，`lsdev` 命令（在 procinfo 包中）能够列出设备使用的通信资源。

`lshw` 程序是上述程序的组合，并能够以分层方式显示发现硬件的详尽描述。应该将该命令的全部输出附加到任何有关硬件支持问题的报告上。

3.5　小结

本章带你概览了 Linux 全貌。讨论了内核和用户空间，回顾了许多常见的 Linux shell 命令，还讨论了进程以及如何管理它们，回顾了用户和组的安全概念，然后讨论了文件系统层次结构标准（FHS），并大体了解了一些在 Kali Linux 上最常见的目录和文件。

要点提示:

- Linux 通常用于指代整个操作系统，但实际上 Linux 本身是指由引导加载程序启动的操作系统内核，该引导加载程序由 BIOS / UEFI 启动。
- 用户空间是指内核之外的一切。在用户空间中运行的程序中，有许多是GNU project[1]的核心功能，其中大部分从命令行运行（一个基于文本的接口，允许输入命令并执行它们，后查看结果）。一个外壳程序（shell）在该接口内负责执行命令。
- 常用命令包括：`pwd`（打印工作目录），`cd`（更改目录），`ls`（列出文件或目录清单），`mkdir`（创建目录），`rmdir`（删除目录），`mv`、`rm` 和 `cp`（分别是移动、删除及复制文件或目录），`cat`（合并或显示文件），`less/more`（对于长内容一次显示一个页面），`editor`（启动文本编辑器），`find`（查找文件或目录），`free`（显示内存信息），`df`（显示磁盘可用空间信息），`id`（显示用户身份以及所属组的

1 http://www.gnu.org

列表），dmesg（查看内核日志）和 journalctl（显示所有可用日志）。

- 可以使用以下命令检查 Kali 系统上的硬件：lspci（列出 PCI 设备）、lsusb（列出 USB 设备）、lspcmcia（列出 PCMCIA 卡）。
- 所谓进程是指一个程序的运行实例，该程序需要存储器来存储程序本身及其操作数据。可以使用以下命令来管理进程：ps（显示进程）、kill（结束进程）、bg（发送进程到后台）、fg（将后台进程转到前台）和 jobs（显示后台进程）。
- 类 UNIX 系统是多用户的。它们支持多个用户和组，并允许根据权限控制用户的操作。可以使用多种命令来管理文件和目录权限，包括：chmod（更改权限）、chown（更改所有者）和 chgrp（更改组）。
- 与其他 Linux 发行版一样，Kali Linux 的组成与文件系统层次结构标准（FHS）一致，这让其他 Linux 发行版的用户能够轻松地使用 Kali。
- 按照惯例，应用程序配置文件存储在主目录下，在以点号（.）开头的隐藏文件或目录中。

现在，我们已经掌握了 Linux 的基础知识，下一章我们将安装并运行 Kali Linux！

练习题

练习1

1. 使用 file 命令观察/dev/目录中内核导出的设备文件。尝试/dev/sda*和/dev/snd/*。

练习2——任务控制

1. 打开终端并运行 ping –I 10 localhost &。
2. 下一步，打开终端并运行 ping –I 10 127.0.0.1 &。
3. 杀掉“localhost”的 ping 进程。
4. 现在杀掉“127.0.0.1”的 ping 进程。

练习3——检索文件

1. 尝试使用能够打印出内核消息缓冲的 dmesg 命令。这个命令的输出通常包含了设备

驱动所产生的消息。

2. 使用 `find` 命令找到文件系统中的 rockyou.txt.gz 文件。
3. 使用 `locate` 命令找到文件系统中的 rockyou.txt.gz 文件。
4. 哪个命令更快：`find` 还是 `locate`？为什么？
5. 你能够想办法测试出一个命令从开始执行到执行结束的准确时间吗？

练习4——枚举硬件

使用 `lspci`、`dmesg` 和其他日志应用程序，发现你的 Kali 主机上的以下信息：

1. Kali 主机的 CPU 类型。
2. 网卡的类型、生产厂商和型号。
3. 显卡的类型、生产厂商和型号。
4. 正在运行的内核版本。
5. 可用的剩余内存容量。
6. 可用的剩余硬盘容量。

练习5——使用硬件

1. 在运行的 Kali 实例中插入 USB 存储器。
2. 确认设备名并连接此硬件。
3. 在运行的 Kali 实例中插入 USB 无线网卡。
4. 确认无线网卡的芯片和型号。

思考题

1. /dev/urandom 是什么类型的设备？
2. 在哪里能够找到服务的配置文件？

第4章　安装Kali Linux

内容

- 最小安装要求
- 手把手教你如何安装到硬盘上
- 无人值守安装
- ARM 平台安装
- 安装疑难解答
- 小结

在本章中，我们将重点介绍 Kali Linux 的安装过程。首先，讨论最小安装要求（4.1 节），以确保真实或虚拟系统已经准备好进行指定类型的安装。然后，我们将详细走一遍安装过程（4.2 节），让你了解如何进行非加密方式安装，以及如何安装一个更安全的加密文件系统。还将讨论预设置（preseeding），通过它能够实现无人值守安装（4.3 节）。而且还将展示如何在各种 ARM 设备（4.4 节）上安装 Kali Linux，从而将 Kali 功能扩展到桌面之外。最后，将展示如何应对非常罕见的安装失败情况（4.5 节）。

4.1　最小安装要求

Kali Linux 的最小安装要求取决于要安装哪些模块。最低端配置的情况，可以将 Kali 设置为无桌面的基本 Secure Shell（SSH）服务器，至少使用 128 MB 内存（推荐 512 MB）和 2 GB 磁盘空间。高端配置的情况，如果选择安装默认的 GNOME 桌面和 kali-linux-full 包，则应该至少准备 2048 MB 内存和 20 GB 磁盘空间。

除了内存和硬盘要求之外，计算机 CPU 需要至少支持 amd64、i386、armel、armhf 或 arm64 中的一种架构。

4.2　手把手教你如何安装到硬盘上

在本章中，我们假设你已有可启动的 USB 驱动器或 DVD 光盘（有关如何准备此类驱动器的详细信息，请参阅 2.1.4 节的内容），你将通过它们引导启动安装过程。

4.2.1　非加密安装

首先，我们来看一个使用未加密文件系统的标准 Kali 安装。

引导并启动安装程序

如图 4.1 所示，一旦 BIOS 从 USB 驱动器或 DVD-ROM 开始启动，会出现 Iso Linux 引导菜单。在这个阶段，Linux 内核尚未加载，此菜单允许选择要启动的内核，并输入传递给它的参数。

对于标准安装，只需要使用箭头键选择 Install，或选择图形模式安装（Graphical install），然后按回车键启动安装过程的其余部分。

每个菜单项都隐藏了一个特定启动命令行，可以在验证入口引导之前按下 Tab 键，然后根据需要进行配置。

图4.1　启动画面

一旦启动，安装程序将引导你一步步完成安装过程。下面我们将详细介绍每一个步骤，

包括标准 Kali Linux DVD-ROM 的安装，以及看起来稍有不同的 mini.iso 安装。我们还将讨论图形模式的安装，其与传统文本模式安装的唯一区别是外观不同，两种安装程序中涉及的问题和选项都是相同的。

选择语言

如图 4.2 所示，安装程序起始界面是英文的，但第一步允许选择后续安装过程显示的语言。此语言选项会影响后续阶段（特别是设置键盘布局时）中定义的默认选项。

使用键盘操作　安装过程中的一些步骤要求你输入一些信息。这些屏幕中有一些区域能够获取焦点（如文本框、复选框、列表框以及 OK 按钮和 Cancel 按钮等），可以使用 Tab 键在这些控件上切换焦点。

在图形安装模式下，你可以像在其他图形界面中一样使用鼠标去控制。

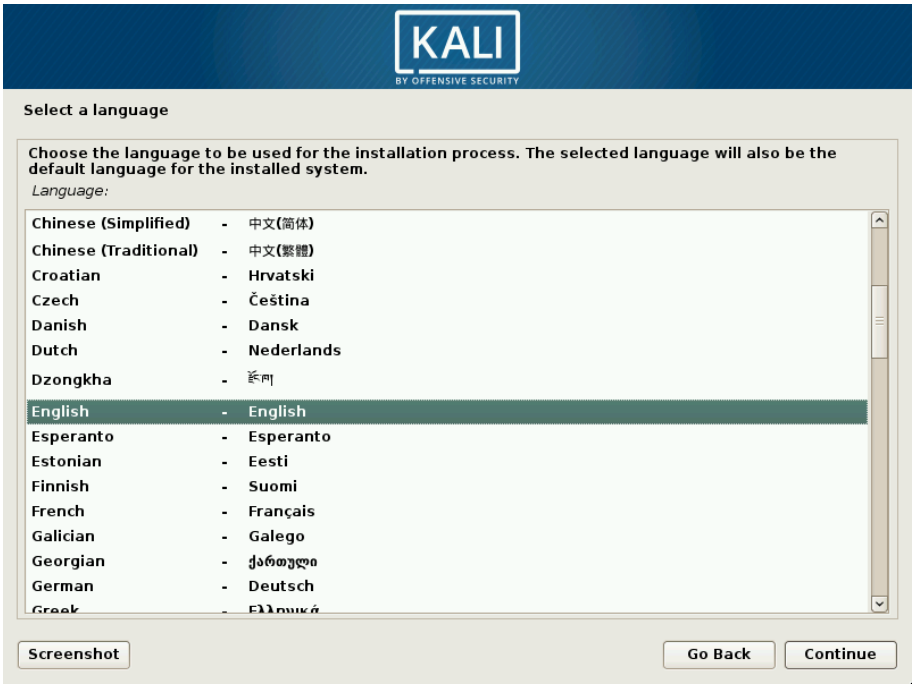


图4.2　选择语言

选择国家

第二步（如图 4.3 所示）选择你的国家。结合所选语言，该信息使安装程序能够提供最合适的键盘布局。这也将影响时区配置。在选择美国（United States）后，系统会推荐使用标准QWERTY键盘，并会显示一个时区选择界面供安装人员设置 [1]。

图4.3　选择国家

选择键盘布局

推荐使用的美式英文键盘对应常用的 QWERTY 布局，如图 4.4 所示。

1 译者注：中国通常使用和美国相同的键盘布局，这一步也推荐选择美国（United States）。

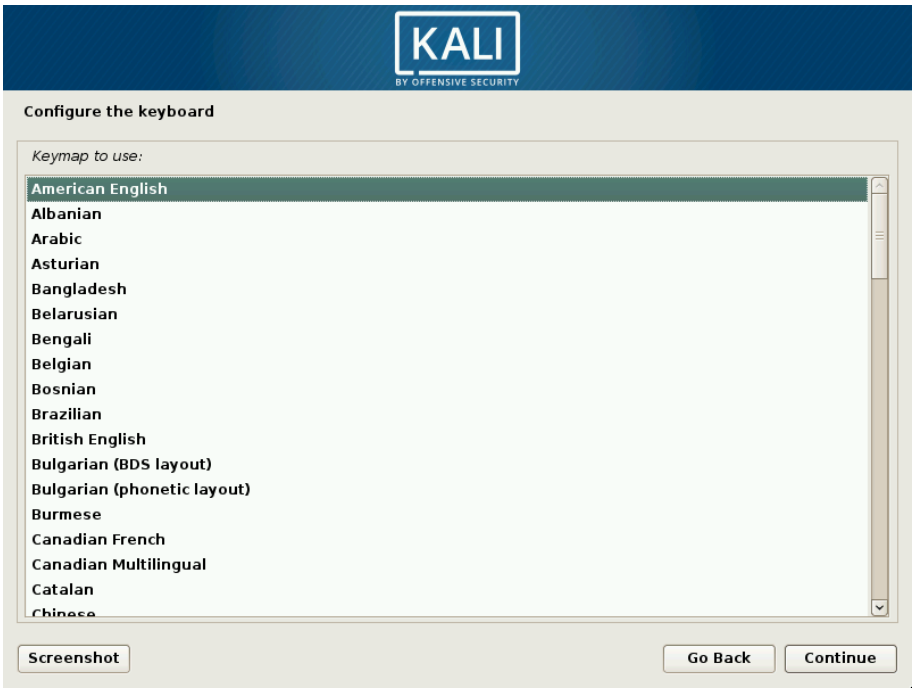


图4.4　配置键盘

硬件检测

在绝大多数情况下，硬件检测步骤完全自动化。安装程序会检测硬件，并尝试识别出引导设备以便访问其内容。它加载那些与检测到的各种硬件组件相对应的模块，然后挂载引导设备以便读取它。上述步骤完全包含在引导设备的启动镜像中，它是一个大小限定的文件，当引导设备启动时，它会被加载到内存中。

载入组件

当引导设备中内容可用后，安装程序将加载所有所需文件，以便继续工作。这包括用于其他硬件（特别是网卡）的驱动程序，以及安装程序的所有组件。

检测网络硬件

在此步骤中，安装程序将尝试自动识别网卡，并加载对应驱动。如果自动检测失败，可以手动选择需要加载的驱动。如果上述步骤都失败了，还可以从可移动设备加载特定模块，这种解决方案通常只有在驱动程序不包含在标准 Linux 内核中时才需要使用，比如需要从其

他地方（例如制造商网站）下载这些驱动。

对于网络安装来说（例如从 mini.iso 引导安装），这一步骤必须确保成功，因为 Debian 软件包必须从网络上下载。

配置网络

为了尽可能使过程自动化，安装程序将尝试使用动态主机配置协议（DHCP）（用于 IPv4 和 IPv6）和 ICMPv6 的邻居发现协议（NDP）（用于 IPv6）进行自动网络配置，如图 4.5 所示。

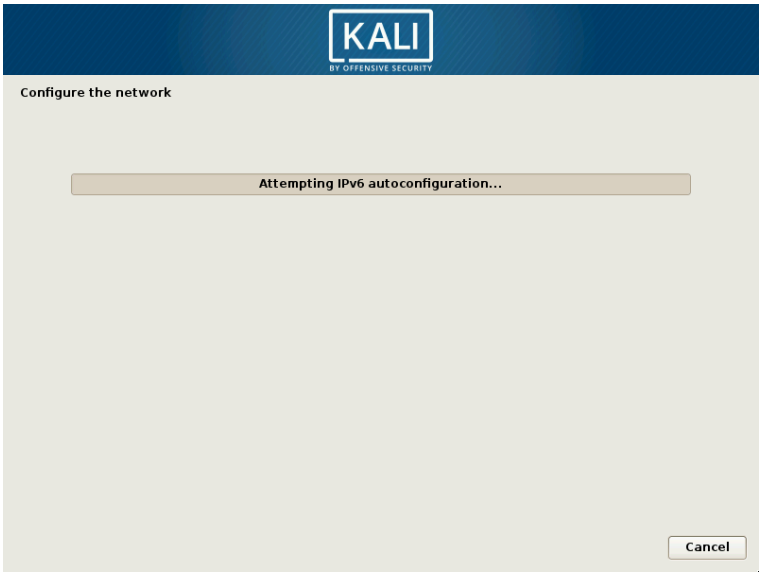


图4.5　自动配置网络

如果自动配置失败，则安装程序会提供一个额外选项：再次尝试使用 DHCP 配置、通过声明机器名称尝试 DHCP 配置，或设置静态网络配置。

如果使用最后一个选项设置静态网络配置，则需要提供 IP 地址、子网掩码、网关 IP 地址、机器名称和域名。

不使用 DHCP 的情况下的网络配置	如果不想使用本地网络中运行的 DHCP 服务器，而希望手工指定一个静态 IP 地址，则可以在启动时添加 `netcfg/use_dhcp=false` 选项。在启动菜单项上按 Tab 键，并输入上述设置后再按回车键继续。

Root口令

安装程序会自动创建超级用户 root 账户，因此会提示输入 root 口令（如图 4.6 所示）。安装程序还要求确认口令，以防止任何输入错误，一旦设置了错误口令，后面很难恢复。

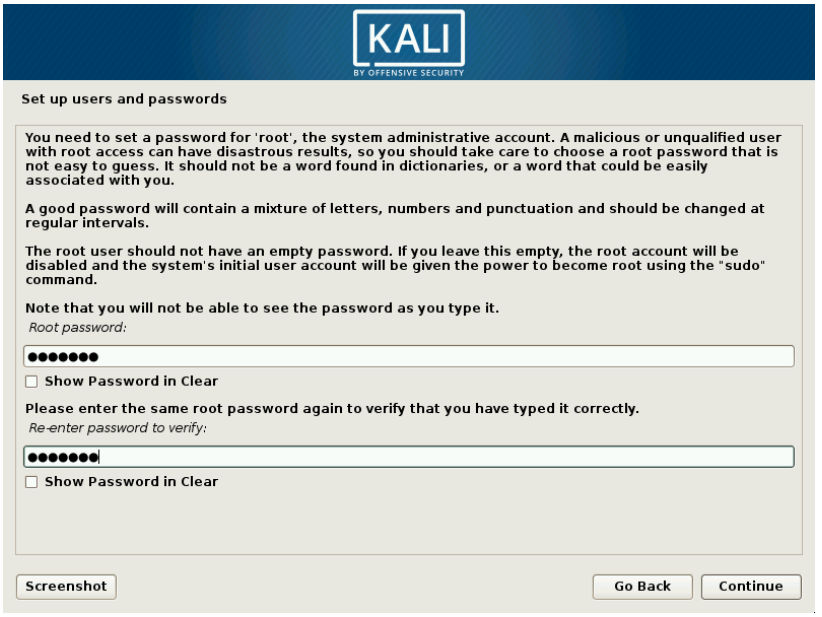


图4.6　Root口令

管理员口令　Root 用户的口令应当足够长（8 个字符或更多）以让人无法猜解，攻击者们会使用自动化工具和常见的弱口令，对连接在互联网上的计算机进行口令攻击。有时候攻击者会实施字典攻击，使用大量单词和数字的组合作为口令进行尝试。避免使用孩子和父母的名字、出生日期这些很容易被猜解的字符作为口令。

其他用户口令也应当遵循这一原则，不过比起管理员用户，一个不具备管理员权限的用户被攻击者获取后造成的后续危害和影响会小得多。

如果你没有设置一个复杂口令的灵感，可以尝试使用口令生成器，例如 `pwgen`（可以在同名的软件包中找到，Kali 的基础安装已包含这个工具）。

设置时钟

如果网络可用，系统内部时钟将会自动从网络时间协议（NTP）服务器更新。这很有好处，因为它能够确保日志上的时间戳从第一次引导开始就是正确的。

如果你的国家跨越多个时区，系统将要求你选择要使用的时区，如图 4.7 所示。

图4.7　时区选择

检测磁盘和其他设备

此步骤将自动检测可能安装 Kali 的硬盘驱动器，每个硬盘驱动器将在下一步要进行的分区中显示。

分区

分区是安装中不可或缺的一步，其主要任务是，根据计算机用途和系统运行的要求，将硬盘驱动器可用空间划分为不相关的区域（分区）。分区同时还会指定使用的文件系统。所有这些决定都会对性能、数据安全和服务管理产生影响。

对于新用户而言，决定如何分区通常很困难。但是，建立 Linux 文件系统和分区，包括虚拟内存（或称为交换分区）是必须的，它们构成了整个系统的基础。如果已经在机器上安装了另一个操作系统，并且希望两者共存，则此任务可能会变得复杂。在这种情况下，必须确保不更改其他操作系统的分区，或者在不会导致数据损坏的情况下，调整其他操作系统分区的大小。

大多数用户更喜欢使用向导模式，它使用起来更简单，这种模式能够向用户提供推荐的

分区参数，并提供每个步骤的详细建议。高级用户喜欢手动模式，以便实现更高级的配置。两种模式有一部分功能是一致的。

使用向导分区：分区工具的第一个屏幕（如图 4.8 所示）显示了向导和手动分区模式的入口点。Guided-use entire disk（向导 - 使用整个磁盘）是最简单和最常见的分区方案，它将整个磁盘分配给 Kali Linux。

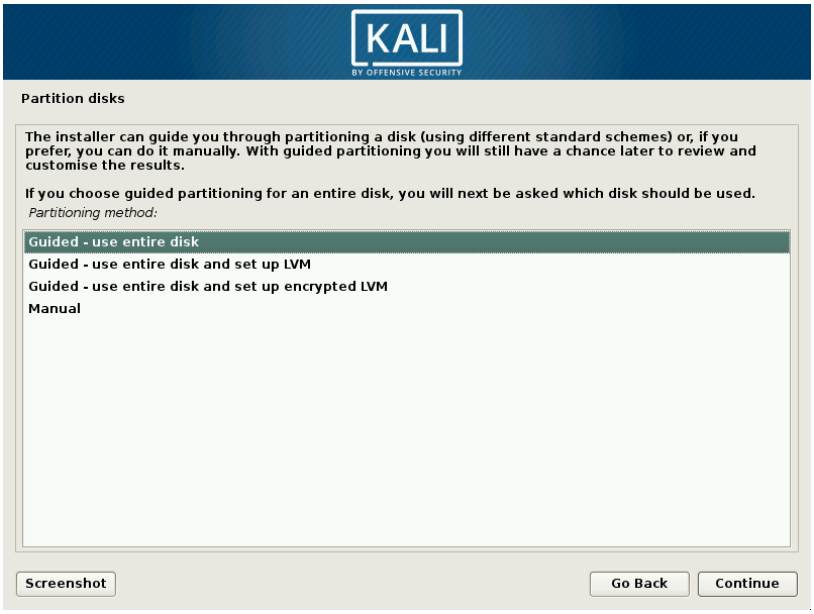


图4.8　分区模式选择

接下来的两个选择使用逻辑卷管理器（LVM）来设置逻辑的、可选加密的分区（代替物理设置）。本章稍后将讨论 LVM 和加密。

最后一个选项会启动手动分区，这样可以实现更高级的分区方案，例如将 Kali Linux 与其他操作系统一起安装。我们将在下一章讨论手动安装模式。

在这个例子中，我们将整个硬盘分配给 Kali，所以我们选择 Guided-use entire disk，然后进行下一步。

下一个屏幕（如图 4.9 所示）允许选择准备安装 Kali 的磁盘（例如，Uirtual disk 1（vda） - 32.2 GB Virtio Block Device）。选择后，分区向导将继续进行下一步。此选项将清除此磁盘上的所有数据，所以要谨慎选择。

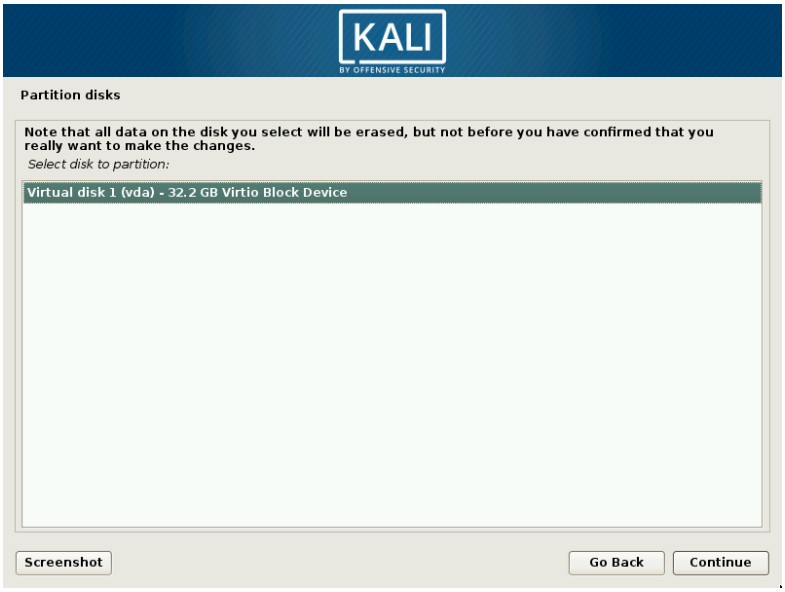


图4.9　选择用于分区的磁盘

接下来，分区向导工具提供三种对应于不同用途的分区方法，如图 4.10 所示。

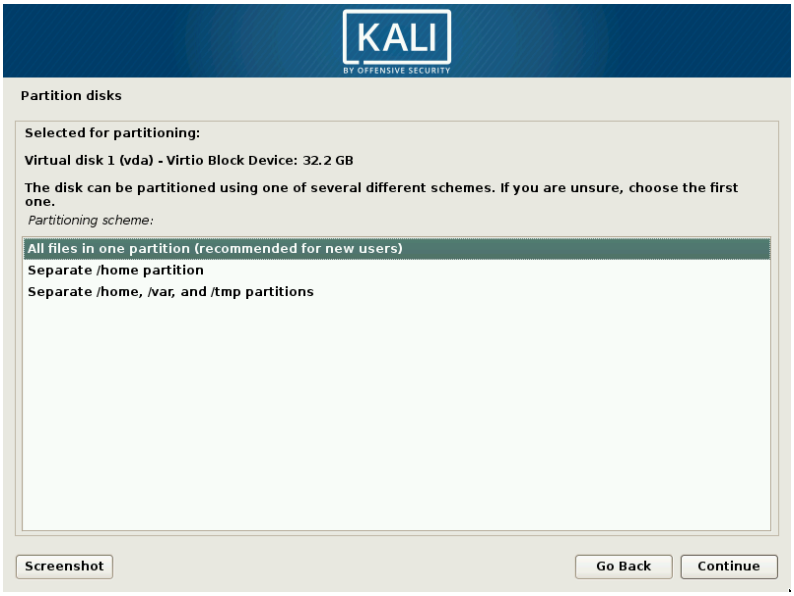


图4.10　分区分配向导

第一种方法为 All files in one partition（所有文件在一个分区中）。整个 Linux 系统树存储在单个文件系统中，对应于根（“/”）目录。这种简单而强大的分区方案对于个人或单用户系统非常有效。尽管名称叫“一个分区”，但是实际上系统仍会创建两个分区：第一个分区存放整个系统；第二个分区用于虚拟内存（或“交换分区”）。

第二种方法为 Separate/home partition（将/home 分离为单独分区），与第一种方法类似，但是该方法将文件层次结构拆分为两个：一个分区包含 Linux 系统（/）；第二个分区包含“主目录”（通常存储用户数据，指在/ home /下的所有目录、子目录和文件）。这种方法的好处在于，如果必须重装系统，那么用户数据很容易被保留下来。

最后一个分区方法称为 Separate/home,/var,and /tmp partitions（将/home、/var 和/tmp 分离为单独的分区），这适用于服务器和多用户系统。它将文件树划分为多个分区。除了根（/）和用户账户（/home/）分区之外，它还有服务器软件数据（/var/）和临时文件（/tmp/）分区。这种方法的好处在于，终端用户无法通过消耗掉所有可用硬盘空间来锁定服务器（只能填满/tmp/和/home/）。同时，守护进程的数据（特别是日志）无法阻塞系统的其余部分。

选择分区类型后，安装程序将你在向导中作出的全部选择作为摘要在屏幕上显示出来（如图 4.11 所示）。可以通过选择某个分区，来单独修改这个分区。例如，如果当前文件系统标准（ext4）不合适，则可以选择另一个文件系统。然而，在大多数情况下，建议分区是可靠的，可以通过选择 Finish partitioning and write changes to disk（完成分区并将更改写入磁盘），来接受这些配置。由于这将会擦除所选磁盘的内容，所以还是要提醒你谨慎选择。

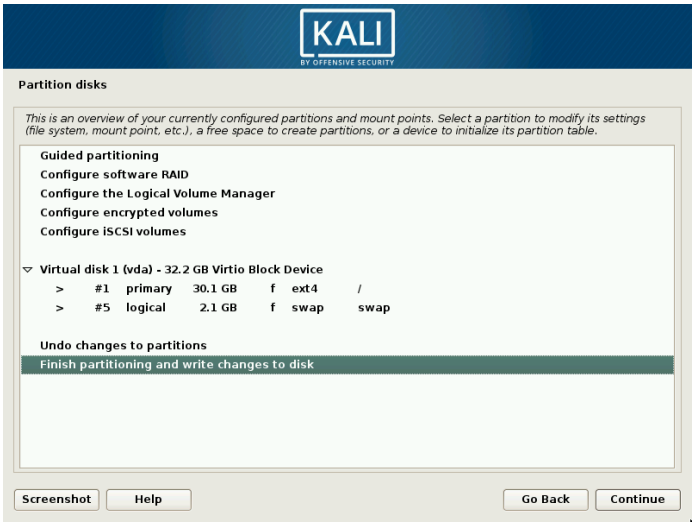


图4.11　验证分区

手动分区：在“为磁盘分区”屏幕（如图 4.8 所示）中选择手动模式，该模式将给予你极大的灵活性，允许选择更高级的配置，你可以详细设置每个分区的目的和大小。例如，此模式允许将 Kali 与其他操作系统一起安装，启用基于软件的磁盘冗余阵列（RAID），来保护数据免受硬盘故障的影响，并可安全调整现有分区大小，且不会丢失数据。

收缩一个 Windows 分区　为了让同一个已存在的操作系统一同去运行 Kali Linux（Windows 或其他），需要提供一块未使用的独立磁盘空间给 Kali。在很多情况下，这意味着需要收缩一个已存在的分区，然后重新利用它的剩余空间。

如果你正在使用手工分区模式，安装程序可以帮助你很轻松地收缩一个分区。你只需要选择这个 Windows 分区，并输入它的新大小就好了（这对于 FAT 和 NTFS 分区都有效）。

如果你对在现有数据系统上安装 Kali 的经验不足，使用此方法时一定要特别小心，因为很容易出错从而导致数据丢失。

除了不包括创建任何新分区外，手动安装过程中的第一个屏幕实际上与图 4.11 所示的屏幕相同。是否添加新的分区取决于你。

首先，你将看到一个进入分区向导的选项，随后会询问几个配置选项。接下来，安装程序将显示可用磁盘、分区以及尚未分区的全部可用空间。可以像往常一样选择每个显示元素，并按回车键。

如果磁盘是全新的，你可能需要创建一个分区表，通过选择磁盘就能够完成这一步操作。完成后，可以看到这块磁盘的可用空间。

如果要使用这块可用空间，选择它并按回车键，安装程序会提供在这个空间中创建分区的两种方法。

第一个条目将创建由你指定特征（包括大小）的单个分区。第二个条目将使用所有可用空间，并将在分区向导帮助下创建多个分区（参见“分区向导”小节内容）。当想要将 Kali 与另一个操作系统一起安装但又不想操心分区布局细节时，此选项非常有用。最后一个条目将显示可用空间开始和结束的柱面/磁头/扇区编号。

当选择 Create a new partition（创建新分区）时，你将进入手动分配分区步骤。安装程序提示输入分区大小。如果磁盘使用 MSDOS 分区表，则可以选择创建主分区或逻辑分区（必须了解的是，你只能拥有四个主分区，但可以有更多逻辑分区。包含`/boot`和内核的分区必须是主分区，逻辑分区驻留在扩展分区中，扩展分区占用四个主分区之一）。完成这些操作以后，将看到通用的分区配置屏幕（如图 4.13 所示）。

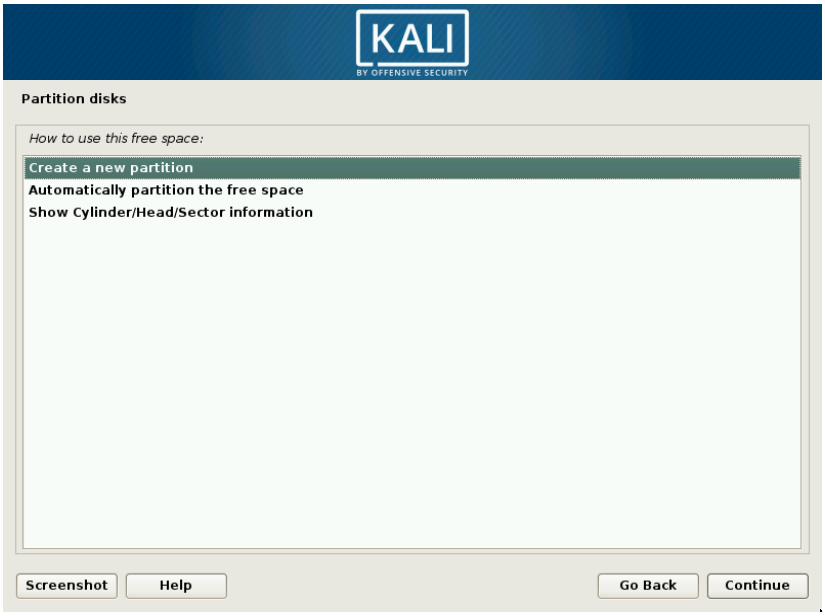


图4.12　在可用空间创建分区

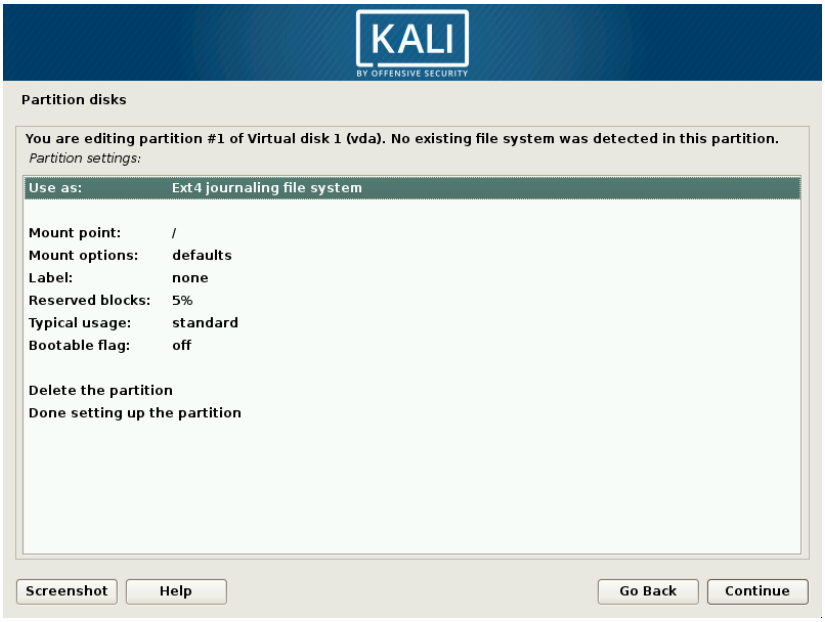


图4.13　分区配置屏幕

下面总结一下手动分区的功能，我们来看看你可以用新的分区做什么。

- 格式化新分区，并通过选择一个挂载点将其包含在文件树中。挂载点是在所选分区上容纳另一个文件系统内容的目录。因此，挂载在/home/上的分区传统上用于包含用户数据，而“/”被称为文件树的根，并且是实际托管 Kali 系统分区的根。
- 将其用作交换分区。当 Linux 内核缺少足够可用内存时，它会将 RAM 非活动部分存储在硬盘上的特殊交换分区中。得益于虚拟内存子系统，这一过程对应用程序来说是透明的。为了模拟额外内存，Windows 使用直接包含在当前文件系统中的交换（分页）文件，而 Linux 使用一个专门分区，称为交换分区。
- 使其成为“加密物理卷”，以保护特定分区中数据的机密性。这一过程在安装向导中能够自动化完成。有关详细信息，请参见 4.2.2 节的内容。
- 使其成为“LVM 物理卷”（本书未涵盖）。请注意，当设置加密分区时，分区向导会自动使用此功能。
- 将其用作 RAID 设备（本书未涵盖）。
- 选择不使用分区，并保持不变。

单击完成后，既可以通过选择 Undo changes to partitions（撤消更改分区），退出手动分区，也可以通过选择 Finish partitioning and write changes to disk（完成分区并将更改写入磁盘）来改写磁盘（如图 4.11 所示）。

复制活动镜像

接下来不需要任何用户的交互，活动镜像内容将被复制到目标文件系统，如图 4.14 所示。

配置包管理器（APT）

为了能够安装附加软件，需要配置 APT 并告知在哪里可以找到 Debian 软件包。在 Kali 系统中，由于我们强制镜像使用 http.kali.org，因此这个步骤多是非交互的。只需确认是否要使用此镜像（如图 4.15 所示）即可。如果不使用它，将无法使用 APT 安装附加软件包，除非稍后配置软件包存储库。

如果想使用一个本地镜像代替 http.kali.org，则可以使用如下语法，在引导屏幕内核命令行传递其名称：`mirror/http/hostname=my.own.mirror`。

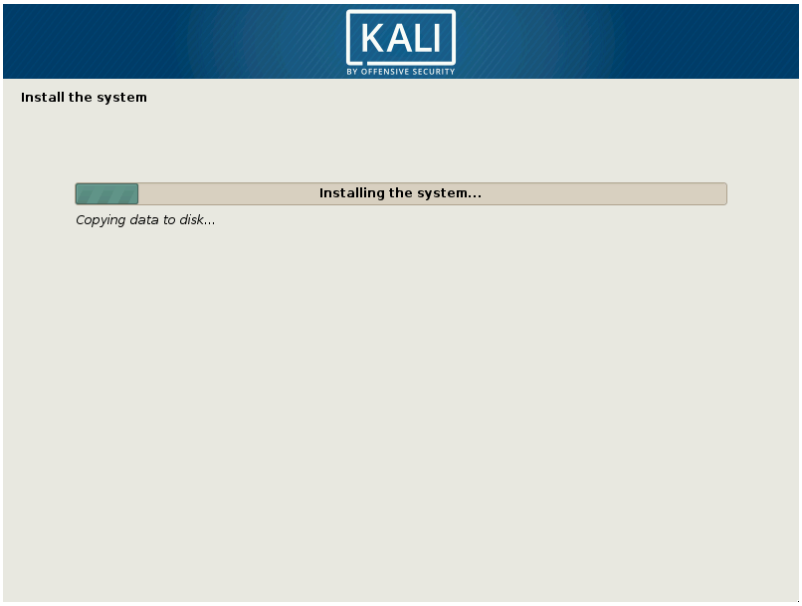


图4.14　从活动镜像复制数据

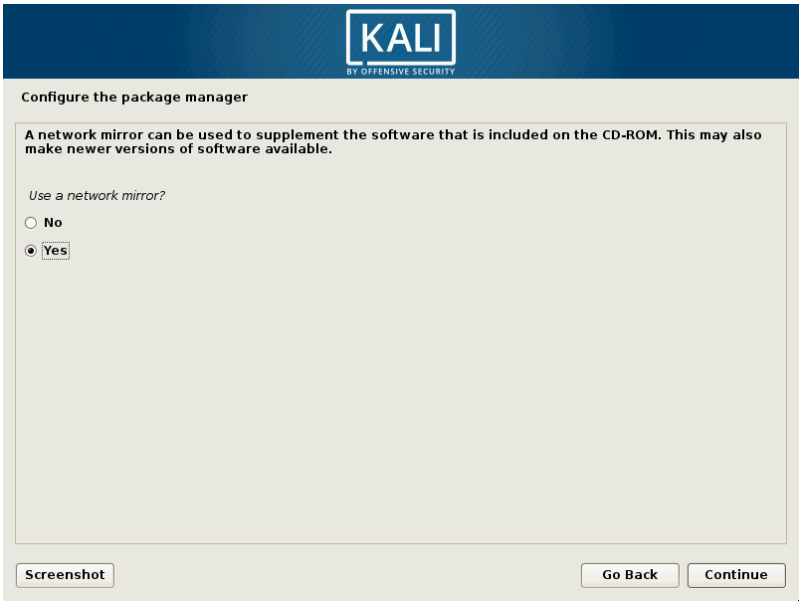


图4.15　使用一个网络镜像

最后，程序询问是否使用 HTTP 代理，如图 4.16 所示。HTTP 代理是转发用户 HTTP 请求的服务器。它有时通过保留已经传输过的文件副本来加速下载（我们称缓存代理）。在某些网络情况下，它可能是访问外部 Web 服务器的唯一方法，这样只有在安装过程中正确填写此字段，安装程序才能下载 Debian 软件包。如果不提供代理地址，安装程序将尝试直接连接到 Internet。

接下来，Packages.xz 和 Sources.xz 文件将会被自动下载，以更新 APT 能够识别的软件包列表。

图4.16　使用一个HTTP代理

安装GRUB引导加载程序

引导加载程序（boot loader）是 BIOS 启动的第一个程序。它会将 Linux 内核载入内存中并执行它。启动引导程序通常会提供一个菜单，该菜单允许你选择要加载的内核或要引导的操作系统。

基于其技术优势，GRUB 是 Debian 安装的默认启动加载程序。它适用于大多数文件系统，因此在每次安装新内核后都不需要更新它，因为它在引导期间会读取配置，并找到新内核的确切位置。

应该将 GRUB 安装到主引导记录（MBR），除非你已经安装了另一个 Linux 系统，而且

它知道如何启动 Kali Linux。如图 4.17 所示，修改 MBR 会导致依赖于它的未识别操作系统无法启动，直到修复 GRUB 配置为止。

在此步骤（如图 4.18 所示）中，必须选择要安装 GRUB 的设备，应该是你当前的启动驱动器。

图4.17　在硬盘上安装GRUB引导加载程序

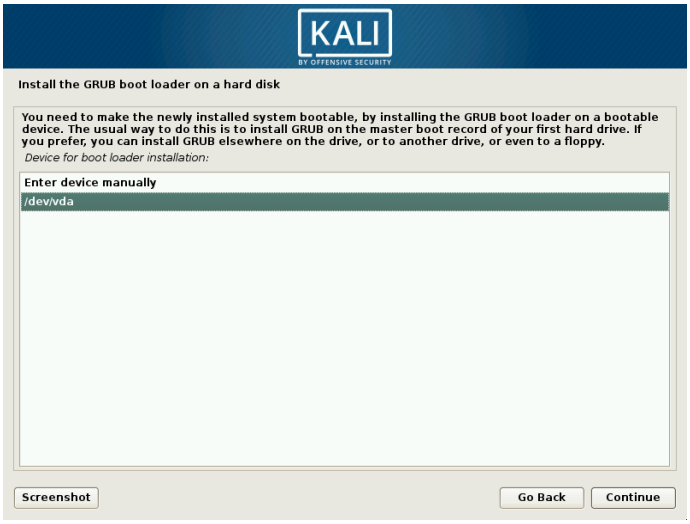


图4.18　引导加载程序的安装设备

在默认情况下，GRUB 提供的启动菜单显示所有已安装的 Linux 内核，以及检测到的其他操作系统。这就是为什么应该接受在主引导记录中安装它的建议。如果最近安装的内核出现故障或者硬件兼容性欠佳，可以从旧的内核版本启动系统。因此，建议保留曾安装的数个较旧的内核版本。

当心：引导加载程序与多重引导

在上面介绍的情况中，安装过程会探测计算机上已安装的操作系统，并自动将相应条目添加到启动菜单上。但是，并不是所有安装程序都会这样做。

特别是，如果在此之后安装（或重新安装）Windows，启动加载程序会被擦除。Kali 虽然还保留在硬盘上，但却无法再从引导加载程序中启动了。这时候必须重启 Kali 安装程序，并使用 `rescue/enable=true` 内核参数重新安装引导加载程序。这些操作在 Debian 安装手册中有相应描述。

➨ `https://www.debian.org/releases/stable/amd64/ch08s07.html`

完成安装并重启

现在安装完成了，程序会要求从光驱中移除 DVD-ROM（或拔下 USB 驱动器），以便在安装程序重启系统后，计算机可以启动到新的 Kali 系统（如图 4.19 所示）。

最后，安装程序将进行一些清理工作，例如删除专门用于创建实时环境（Live environment）的软件包。

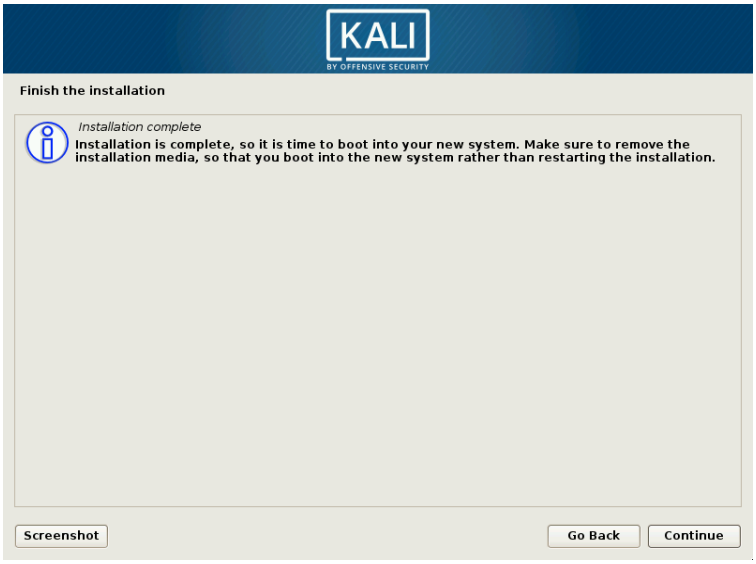


图4.19　安装完毕

4.2.2 在一个完全加密的文件系统上安装

为了保证数据机密性，可以设置加密分区。如果笔记本电脑或硬盘驱动器丢失或被盗，这样做能够保护你的数据。无论是使用向导模式还是手动模式，分区工具都可以在这个过程中帮到你。

分区向导将会组合使用两种技术：用来加密分区的 Linux 通用密钥设置（LUKS）和用来管理动态存储的逻辑卷管理（LVM）。这两个功能也可以在人工分区模式下管理和配置。

LVM介绍

我们先来讨论 LVM。用 LVM 术语来说，虚拟分区（virtual partion）就是逻辑卷（logical volumn），是卷组（volumn group）的一部分，或是多个物理卷（physical volumn）的组合。物理卷是实际分区（或由其他抽象技术导出的虚拟分区，例如软件 RAID 设备或加密分区）。

由于没有“物理”和“逻辑”分区之间的严格区别，LVM 允许创建跨多个磁盘的“虚拟”分区。好处是双重的：分区大小不再受单个磁盘容量限制，而是受磁盘容量总和限制，你可以随时调整现有分区大小，例如在添加额外磁盘之后。

这种技术非常简单。每个卷（无论是物理卷还是逻辑卷）被分割成相同大小的块，与 LVM 相关联。添加一个新磁盘将创建一个新物理卷，这个新物理卷将提供能够与任何卷组相关联的新块。卷组中的所有分区随即可充分利用这些额外分配的空间。

LUKS介绍

为了保护数据，可以在选择的文件系统下添加一个加密层。Linux（具体指 dm-crypt 驱动程序）使用设备映射创建一个基于底层分区的虚拟分区，该虚拟分区将以加密形式存储数据（得益于 LUKS）。LUKS 实现了加密数据以及加密算法元数据存储的标准化。

加密的交换分区

当使用一个加密分区时，加密密钥将被保存在内存（RAM）中。当计算机（如笔记本电脑）休眠时，这个密钥会和其他内存中的内容一起被复制到硬盘交换分区中。既然能够访问交换分区的任何人（包括技术专家或窃贼）都可以从分区中恢复密钥并解密数据，那么交换分区也需要进行加密保护。

正因为此，如果你尝试将加密分区和未加密交换分区一起使用，安装程序会向你发出警告。

创建加密分区

除了分区步骤（如图 4.20 所示）之外，加密 LVM 的安装过程与标准安装相同。在这一步应选择 Guided - use entire disk and set up encrypted LVM（向导-使用整个磁盘并设置加密的 LVM）。使用这个选项，你将得到一个只有提供密码才能启动或访问的系统。这将会加密并保护磁盘上的数据。

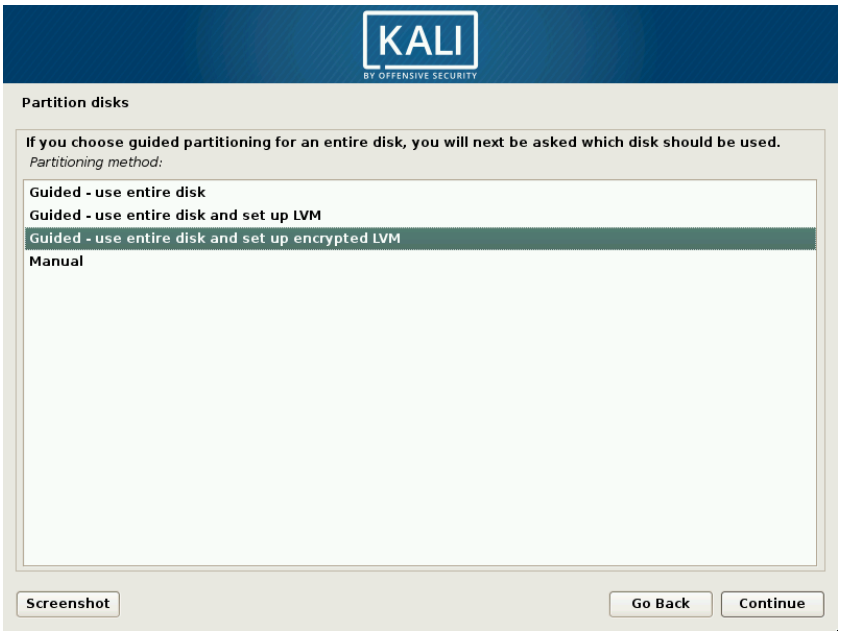


图4.20　加密LVM分区向导

分区向导将自动为存储加密数据分配一个物理分区，如图 4.21 所示。此时，安装程序将在把更改写入磁盘之前发出确认提醒。

新分区将使用随机数据进行初始化，如图 4.22 所示。这使包含数据的区域与未使用区域不可区分，使针对加密数据的检测以及随后攻击变得更加困难。

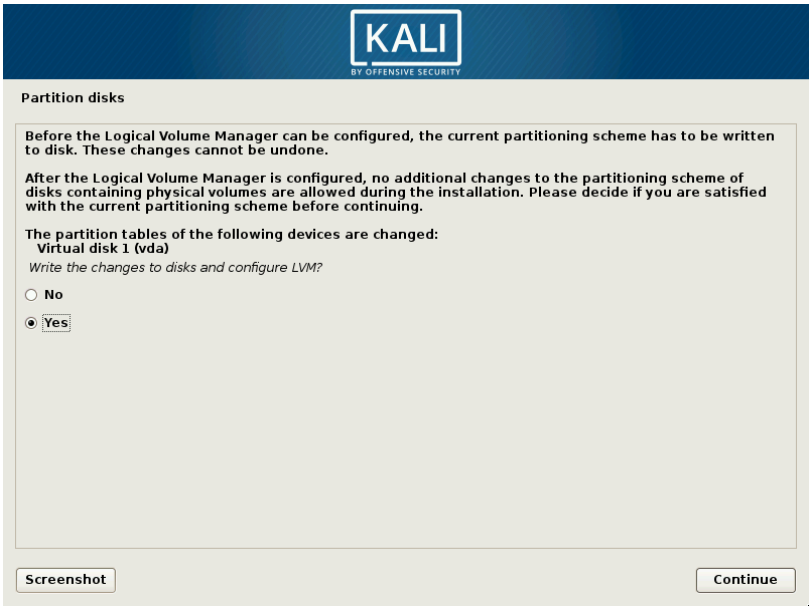


图4.21　确认写入分区表的更改

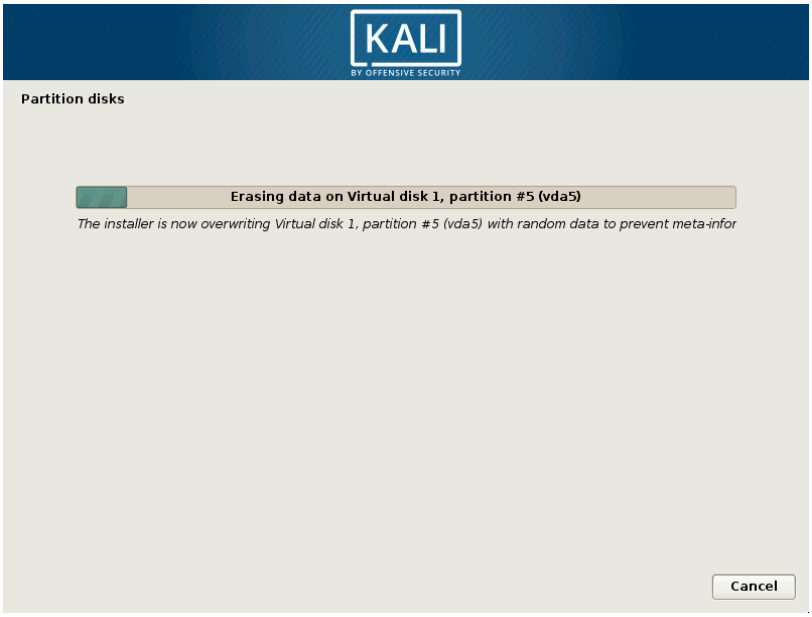


图4.22　从加密分区擦除数据

接下来，安装程序会要求输入加密密码（如图 4.23 所示）。为了查看加密分区内容，需要在每次重启系统时输入此密码。请注意安装程序中的警告：加密系统的强度仅会与此密码的强度一致。

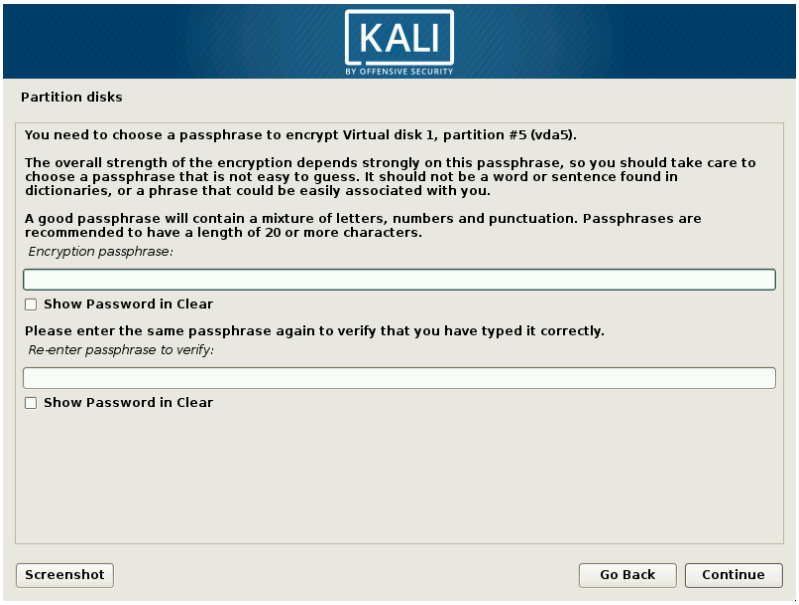


图4.23　输入加密密码

分区工具现在可以访问底层物理分区中的新加密虚拟分区中的数据。由于 LVM 将此新分区用作物理卷，因此可以使用相同的加密密钥（包括交换分区）来保护多个分区（或 LVM 逻辑卷）（请参阅“加密的交换分区”中的内容）。在这里，LVM 并不是用来让扩展存储大小更加容易的，而是更方便地将单个加密分区拆分成多个逻辑卷。

结束加密LVM分区向导

接下来会显示分区方案（如图 4.24 所示），以便根据需要调整设置。

最后，在验证完分区设置后，工具要求确认将更改写入磁盘，如图 4.25 所示。

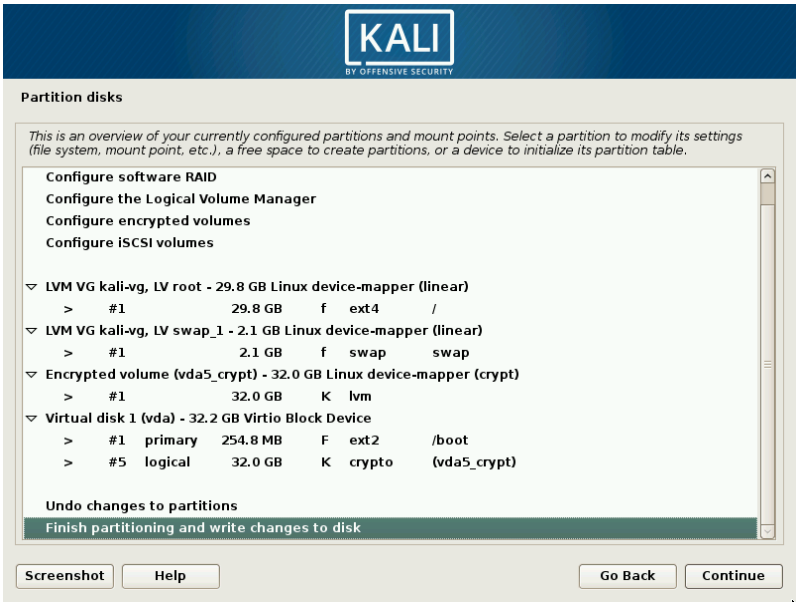


图4.24　验证加密LVM安装的分区

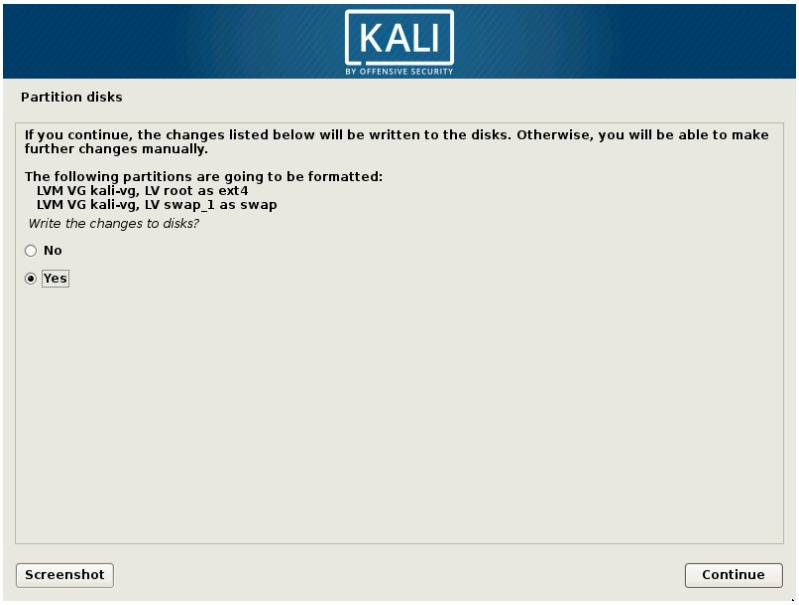


图4.25　确认待格式化的分区

随后安装步骤会像在前面“配置包管理器（APT）”中介绍的一样配置包管理器。

4.3　无人值守安装

Debian 和 Kali 安装程序非常模块化。实际上，它们是在一个接一个地执行许多脚本，这些脚本被打包为称为 udeb（意为 μdeb 或 micro-deb）的微型包。每个脚本都依赖于 debconf（参阅“debconf 工具”），它们与你、用户及预存安装参数进行交互。因此，安装程序也可以通过 debconf 的预设答案功能，实现预设自动化，这个功能允许为安装中问题预先提供无人值守安装的答案。

4.3.1　预设答案

有多种方法来预设安装程序的答案。每种方法都有自己的优缺点。根据预设发生时间不同，可以使用多种方法去预设问题。

使用启动参数

可以在内核命令行的引导参数中，预设任何一个安装问题的答案，可通过/proc/cmdline 访问。一些引导程序允许以交互方式编辑这些参数（这对于测试是很实用的），但是如果希望永久保留这些更改，则必须修改引导加载程序的配置文件。

可以直接使用debconf问题完整标识符（例如`debian-installer/language=en`），或者可以使用最常见问题的缩写（例如`language=en`或`hostname=duke`）。请参阅Debian安装手册中别名的完整列表[1]。

由于启动引导参数在安装过程一开始就可以使用，并且很早就被处理，所以这种方法对于预设哪些问题并没有什么限制。但是，启动参数的数量限制为 32，而且其中一些位置默认情况下已经被使用。同样需要注意，改变引导加载程序的配置有时会导致难以预料的结果。

在 9.3 节中，我们还将学习生成自己的 Kali ISO 镜像时如何修改 ISO linux 配置。

使用Initrd中的预设文件

可以在安装程序中 initrd 的根目录下添加一个名为 preseed.cfg 的文件（这是用于启动安装程序的 initrd）。通常，这需要重建 debian 安装程序源码包，来生成 initrd 的新版本。但是，live-build 提供了一种简便方法，9.3 节会详细介绍这种方法。

此方法对于你的预设问题也没有任何限制，因为 preseed.cfg 文件在引导后立即可用。在

1 https://www.debian.org/releases/stable/amd64/apbs02#preseed-aliases

Kali 中，我们已经利用此功能自定义了部分 Debian 官方安装程序的行为。

使用引导介质中的预设文件

可以在引导介质（CD 或 USB 存储设备）上添加一个预设文件，这样，在安装程序完成媒介挂载后，也就是在语言和键盘布局问题之后，预设答案即可生效。引导时可以使用 `pressed/file` 参数，指明预设文件位置（例如，从 CD-ROM 安装时为 `/cdrom/preseed.cfg`，或从 USB 存储设备安装时为`/hd-media/preseed.cfg`）。

语言和国家选项无法通过这种方式进行自动应答，因为这时相应硬件驱动器尚未被挂载，预设文件也无法读取。不过好处在于，live-build 可以在生成的 ISO 镜像中放置补充文件，以便于操作（参见 9.3 节的内容）。

使用从网络上获取的预设文件

可以通过 Web 服务器从网络上提供预设文件，然后通过添加引导参数 `preseed/url=http://server/pressed.cfg`（或使用 url 别名）告知安装程序从何处下载该预设文件。

不过，使用此方法时，请记住，必须首先配置好网络。这意味着，与网络相关的 debconf 问题（特别是主机名和域名）及之前所有问题（如语言和国家），都不能用此方法来预设。此方法通常与使用引导参数的方法组合使用。

这种预设方法是最灵活的，你可以在不更改安装介质的情况下更改安装配置。

延迟语言、国家、键盘设置　为了解决在一些环境中无法预设语言、国家和键盘问题的限制，你可以添加一个引导参数 `auto_install/enable=true`（或 `auto=true`）。如果使用这个参数，只有在配置完网络并下载了预设文件后才会询问这几项。

这样做的不利影响是，第一步就是网络配置，安装界面永远都是英文，如果出现问题只能在英文界面下解决，而且会自动配置为 QWERTY 键盘。

4.3.2 创建预设文件

预设文件是纯文本文件，其中每行包含一个 debconf 问题的答案。每一行被空格（或制表符）分隔为四个字段。例如：

- 第一个字段表示问题所有者。例如，“d-i”用于表示与安装程序相关的问题。你也可以看到软件包名称，这表示问题来自 Debian 的软件包（如 `atftpd atftpd/use_inetd Boolean false`）。

- 第二个字段是问题标识符。
- 第三个字段列出问题类型。
- 第四个也就是最后一个字段，包含了预期答案的值。请注意，它必须与第三个字段以一个空格分隔，额外的空格字符会被认为是值的一部分。

编写预设文件最简单的方法是手动安装一次系统。然后，`debconf-get-choices --installer` 命令将会把你提供给安装程序的答案提供给你。你还可以通过 `debconf-get-selections` 命令获取针对其他软件包提问的答案。然而，一个更直接的解决方案是手动编写预设文件，可以从一些例子开始，然后仔细地阅读文档了解其用法。使用这种方法，只有那些其默认答案需要被覆盖的问题才可以被预先设定答案。使用 `priority=critical` 引导参数指示 debconf 只提出关键问题，并对其他问题使用默认选择项。

安装指导附录

在线的 Debian 安装指导在附录中包含了一份如何使用预设文件的详细文档。它同时包含了一个含注释的详细文件，可以作为本地自定义用的模板。

➨ `https://www.debian.org/releases/stable/amd64/apb.html`

➨ `https://www.debian.org/releases/stable/example-preseed.txt`

请注意，上面链接中的文档针对的是 Debian 稳定版，而 Kali 使用了测试版，所以有不少细节会有区别。你还可以参考在 Debian-installer 项目网站上的文档，那些文档可能会更新一些。

➨ `http://d-i.alioth.debian.org/manual/en.amd64/apb.html`

4.4　ARM平台安装

Kali Linux 能够在各种基于 ARM 的设备上运行（例如笔记本电脑、嵌入式计算机和开发板等），但是不能在这些设备上使用传统的 Kali 安装程序，因为这些设备通常对内核或引导加载配置方面具有特殊要求。

为使这些设备更容易被Kali用户使用，Offensive Security开发了脚本，以用于构建各种ARM设备上使用的磁盘镜像 [1]。他们在网站上提供这些镜像的下载：

➨ `https://www.offensive-security.com/kali-linux-arm-images/`

由于提供了这些镜像，在 ARM 设备上安装 Kali 的任务非常简单。以下是基本步骤：

1．下载适用于你 ARM 设备的镜像，并确保校验和与网站上提供的校验和相匹配（参见 2.1.3 节的内容，了解如何执行此操作）。请注意，镜像通常是 xz 压缩的，确保使用 unxz 解

1 https://github.com/offensive-security/kali-arm-build-scripts

压缩它们。

2．根据特定 ARM 设备上可用的存储扩展接口，要求至少 8 GB 的 SD 卡、微型 SD 卡或 eMMC 模块。

3．使用 dd 命令将下载的镜像复制到存储设备。这与将 ISO 镜像复制到 USB 存储设备的过程类似（参见 2.1.4 节的内容）。

```
#dd if = kali-image.img of = / dev / something bs = 512k
```

4．将 SD 卡/eMMC 插入 ARM 设备。

5．启动 ARM 设备并登录（用户名为“root”，密码为“toor”）。如果你的设备不具备或未连接显示屏，则必须找出通过 DHCP 分配的 IP 地址，并通过 SSH 连接到该地址。一些 DHCP 服务器提供工具或 Web 界面来显示当前的 IP 租用状况。如果没有这样的工具，可以使用嗅探器来查找 DHCP 分配的 IP 地址。

6．更改 root 密码并生成新的 SSH 主机密钥，特别是设备将在公共网络上永久运行的情况！这类步骤相对简单，请参见“生成新的 SSH 主机密钥”中的介绍。

7．尽情享用运行了 Kali Linux 的新 ARM 设备吧！

特殊状况与更详细的文档

这些指导是通用的，也能够适用于大多数设备，但总有例外发生。例如，Chromebook 需要设置为开发模式，而一些设备需要按一些特定按键以便从外部媒介启动。

鉴于 ARM 设备日新月异，加上这些设备的标准也经常在变化，所以我们无法在这里指导每一种 ARM 设备的安装。你可以查看由 Offensive Security 支持的 Kali 在线文档中的“ARM 上的 Kali”章节去了解 Kali 所支持的每一种 ARM 设备的信息。

➨ http://docs.kali.org/category/kali-on-arm

4.5 疑难问题解答

安装程序本身相当可靠，但你可能会遇到诸如网络问题、镜像损坏及磁盘空间不足等错误或外部问题。因此，你还需要了解一些排除安装过程中问题的技巧。

当安装程序失败时，它将显示一个毫无用处的屏幕，如图 4.26 所示。

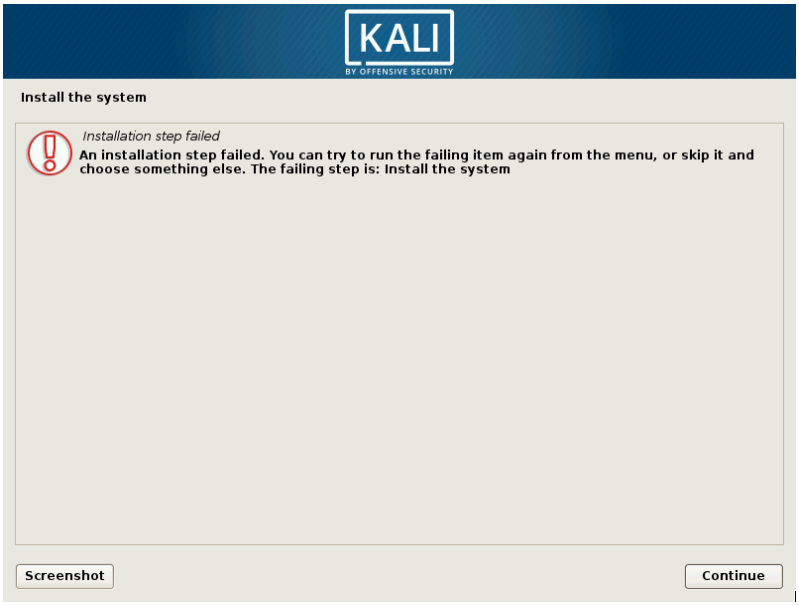


图4.26　安装步骤失败

此时，如果你知道安装程序使用了多个虚拟控制台，这将是一件令人愉快的事。你看到的主屏幕其实是第五个控制台（图形安装程序，Ctrl+Shift+F5 组合键）或第一个控制台（对于文本安装程序，Ctrl+Shift+F1 组合键）。在这两种情况下，第四个控制台（Ctrl+Shift+F4 组合键）会显示正在发生事件的日志，在这里通常可以看到更有用的错误信息，如图 4.27 所显示的那样，表明安装程序已经用完磁盘空间。

第二和第三个控制台（Ctrl+Shift+F2 组合键和 Ctrl+ Shift+F3 组合键）中托管的 shell 可以用来更详细地调查当前情况。大多数命令行工具由 BusyBox 提供，因此功能集相当有限，但通过它们也足以弄清楚可能遇到的大多数问题。

可以在安装程序 shell 里做什么

可以使用 `debconf-get` 和 `debconf-set` 命令查看和编辑 debconf 数据库。这些命令在测试预设答案文件方面非常有用。

可以使用 `cat` 或 `more` 命令查阅任意文件（例如在`/var/log/syslog` 中的完整安装日志）。可以使用 `nano` 命令编辑任意文件，包括安装到系统上的所有文件。一旦分区步骤完成，根文件系统会被挂载在/target 上。

网络访问配置完毕后，可以使用 `wget` 和 `nc`（`netcat`）命令从互联网下载数据或上传数据到互联网。

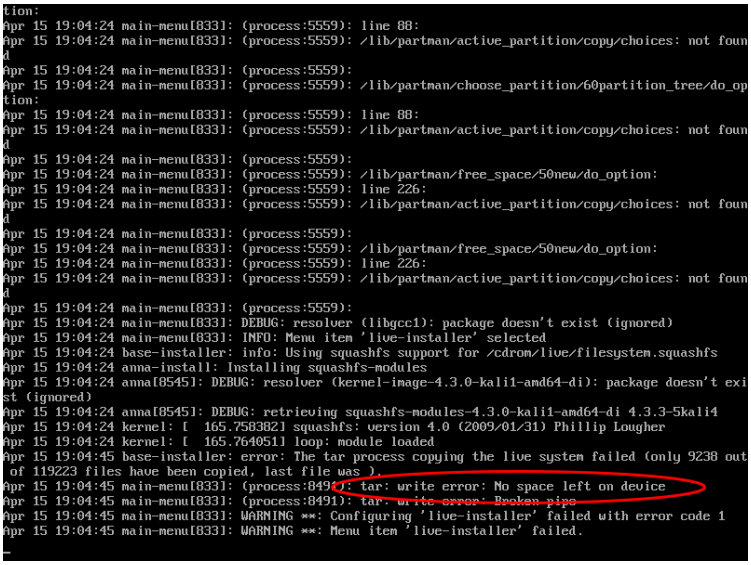


图4.27　安装程序的日志屏幕

在安装程序失败的主屏幕（如图 4.26 所示）上单击 Continue 按钮后，将返回到通常不会看到的屏幕（如图 4.28 所示），这个菜单允许手动一个接一个地启动安装步骤。如果你设法通过 shell 访问解决了问题（恭喜你），则可以重试失败的步骤。

如果无法解决问题，你可能想要存档一份错误报告。该报告应当包含安装程序日志，可以使用主菜单中的 Save debug logs 功能获取安装程序日志。它提供多种导出日志的方法，如图 4.29 所示。

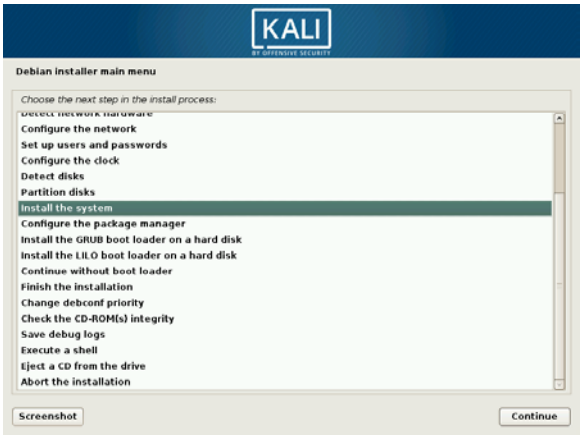


图4.28　安装程序主菜单

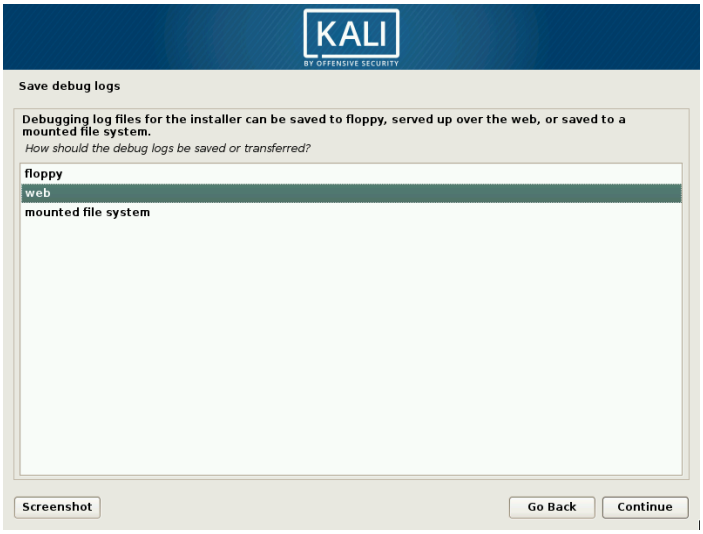


图4.29　保存调试日志（1）

最简便的方法，也是我们推荐的方法，是让安装程序启动托管日志文件的 Web 服务器（如图 4.30 所示）。之后可以从同一网络上的另一台计算机启动浏览器，并下载所有日志文件和由 Screenshot 按钮所截取的每个安装步骤的截图。

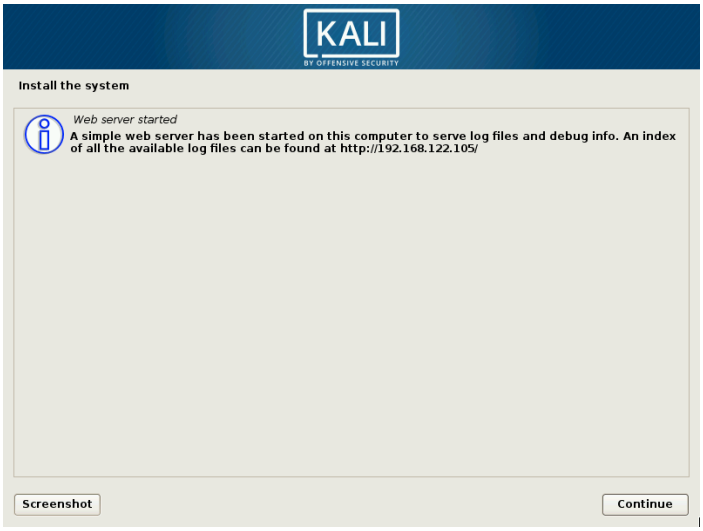


图4.30　保存调试日志（2）

4.6 小结

在本章中，我们主要介绍了 Kali Linux 的安装过程。讨论了 Kali Linux 的最低安装要求，标准和完全加密文件系统的安装过程，允许无人值守安装的预设文件，如何在各种 ARM 设备上安装 Kali Linux，以及在安装失败的罕见情况下如何应对。

要点提示：

- Kali Linux 安装的最低要求是无桌面的基本 SSH 服务器、至少 128 MB RAM（推荐 512 MB）和 2 GB 磁盘空间，高端的 kali-linux-full 元包、至少 2048 MB RAM 和 20 GB 磁盘空间。此外，CPU 必须支持 amd64、i386、armel、armhf 或 arm64 等架构的一种。
- 通过分区和修改引导加载程序，可以轻松地将 Kali 安装为主操作系统、虚拟机或与其他操作系统共存。
- 为了保证数据的机密性，可以设置加密分区。如果笔记本电脑或硬盘驱动器丢失或被盗，这将保护你的数据。
- 安装程序也可以通过 debconf 预设进行自动化安装，这个功能允许为安装程序提出的问题提供无人值守的答案。
- 预设文件（preseed file）是纯文本文件，其中每行都包含一个 debconf 问题的答案。每一行用空格（空格或制表符）分隔为四个字段。可以通过在 initrd 和引导介质上使用预设文件或者使用来自网络的预设文件为引导参数程序预设答案。
- Kali Linux 在各种基于 ARM 的设备上都可以运行，如笔记本电脑、嵌入式计算机和开发板。ARM 安装相当简单。下载相应镜像，将其刻录到 SD 卡、USB 驱动器或嵌入式多媒体控制器（eMMC）模块中，将其插入，启动 ARM 设备，在网络上查找设备，登录并更改 SSH 密码和 SSH 主机密钥。
- 安装失败时，可以使用虚拟控制台（可通过 Ctrl+Shift+功能键访问）中的 `debconf-get` 和 `debconf-set` 命令，读取`/var/log/syslog` 日志文件，或是通过安装程序中的 Save debug logs 功能提交一个日志文件的错误报告来进行调试。

前面我们已经讨论了 Linux 的基础知识和 Kali Linux 的安装，下一章我们将讨论配置，这样你就可以根据需要来量身定制 Kali 了。

练习题

练习1——全加密盘安装Kali Linux

1．一台 Kali 虚拟机的最低资源分配是怎样的？

2．将一个标准的、默认的、全盘加密的 Kali Linux 安装到一个新的虚拟机中。确保网络设置为 NAT 模式。

3．加密用了什么样的技术？

练习2——Kali Linux无人值守安装

1．创建一个新的虚拟机，使用最低的硬件配置。

2．使用 HTTP 服务器（或 HTTPS 服务器）中托管的预配置（preseed）文件完成一次标准的默认安装，可以参考或使用 https://www.kali.org/dojo/preseed.cfg 中的介绍。

3．确保实现安装全程无人值守，你必须预配置地区设置、键盘映射、主机名、域名等项目。

练习3——Kali Linux在ARM设备中的安装

1．如果你有一个树莓派或类似的设备，从 Kali 网站上下载相应的 ARM 镜像，烧录到一张 SD 卡并尝试运行它。

进阶练习

练习4——自定义Kali Linux ARM安装

1．在上个练习中，我们完成了一次标准的 ARM 安装。如你所见，安装得到的系统并不尽如人意。虽然我们未在本书中介绍如何创建一个自定义的镜像，但是这件事情是值得你去尝试的。你可以使用任何支持的 ARM 设备完成本练习，不过我们用的是一台树莓派 3，在开始前请检查官方支持的 ARM 硬件清单。在本练习中我们将创建一个自定义的 Kali ARM 镜

像，包含以下功能：

- 最小的软件包组合。
- 不包含桌面环境。
- 为 eth0 配置静态 IP 地址以便更简单地定位我们的树莓派。
- 安装类似 ifconfig 的工具。
- 启动时自动开启 SSH 服务，并预安装 SSH 密钥。

练习5——Kali Linux ARM chroot

你构建的自定义镜像似乎并不理想，幸运的是，你可以随时修改它。在这个例子中，我们假设你忘了安装一些软件包，比如 net-tools、dnsmasq 和 mlocate。你可以从 Kali 主机使用 `chroot` 命令切换到树莓派的 SD 卡进行必要的更改，而不需要重新安装和重新生成镜像。

第5章　配置Kali Linux

内容

- 配置网络
- 管理 UNIX 用户和组
- 配置服务
- 管理服务
- 小结

在本章中，我们将介绍如何配置 Kali Linux。首先，在 5.1 节中，我们将演示如何使用图形界面以及命令行的方式来进行网络设置。在 5.2 节中，我们将讨论用户和用户组，告诉大家如何创建账号、修改用户、设置密码、禁用账号和管理用户组。最后，在 5.3 节中讨论系统中的各个服务，告诉大家应该如何设置和调用一些常用服务，并将着重介绍三个非常重要的服务：SSH、PostgreSQL 和 Apache。

5.1　网络配置

5.1.1　使用桌面上的NetworkManager

在标准 Kali 系统中，已经安装了 NetworkManager 功能，正如图 5.1 所示，可以通过右上角的 GNOME 控制中心菜单，对网络进行控制和配置。

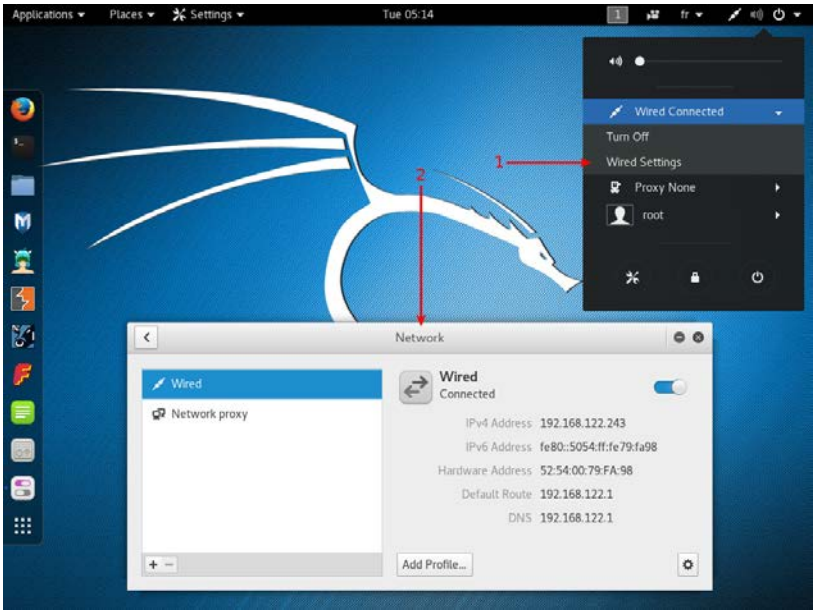


图5.1　网络配置

默认的网络配置通过 DHCP 来自动获取 IP 地址、DNS 服务器以及网关，但是可以通过单击右下角的齿轮，来调整网络配置（例如设置 MAC 地址，切换到静态设置，启用或禁用 IPv6，添加路由路径）。还可以通过创建配置文件，来保存多个有线网络的网络配置，并通过配置文件可以轻松随意地在不同网络之间切换。对于无线网络，它们的设置会被自动绑定其公共标识符（SSID）。

NetworkManager 也能够处理移动网络连接（广域无线网），以及通过调制解调器使用 pppoe 模式的拨号上网。最后还有很重要的一点，它可以通过集成各种专用插件，提供各种类型虚拟专用网络（VPN）的接入功能，如 SSH、OpenVPN、Cisco 的 VPNC、PPTP、Strongswan。为了更好地使用这些功能，我们可以查看 network-manager 扩展包的安装状态；很多扩展包都不是默认安装的。需要使用`-gnome` 命令来对扩展包添加后缀，以便通过图形界面配置和使用。

5.1.2　通过命令行使用ifupdown

如果你不喜欢使用（或无法使用）图形界面，则可以使用已安装的 ifup 和 ifdown 两个工具的 `ifupdown` 软件包来配置网络。这些工具的详细配置可以在

/etc/network/network/ interface 配置文件看到，它们也是系统启动时配置网络的 /etc/init.d/networking 脚本中的核心部分。

我们随时可以通过 ifdown network-device 功能来禁用由 ifupdown 管理的任何一个网络设备。然后可以在/etc/network/interfaces 中修改，并用 ifup network-device 将网络设备（使用新的配置）重新启用。

下面我们来看看 ifupdown 的配置文件中都有些什么。可以看到其中有两个主要部分：第一部分是 auto network-device，告诉 ifupdown 模块，一旦可用时就自动配置该网络接口；第二部分为 iface network-device inet/inet6 type，用于配置指定的接口。举个例子，一个简单的 DHCP 配置如下所示：

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp
```

需要注意的是，针对回环（loopback）设备的特殊配置需要始终保留在这个配置文件中。如果要给网卡配置一个固定的静态 IP 地址，必须提供更多的细节，如 IP 地址、子网掩码、网关 IP 等：

```
auto eth0
iface eth0 inet static
  address 192.168.0.3
  netmask 255.255.255.0
  broadcast 192.168.0.255
  network 192.168.0.0
  gateway 192.168.0.1
```

如果要接入无线网络，必须要安装 wpasupplicant 软件包（Kali 中默认已安装），它提供了可以在 /etc/network/interfaces 中使用的许多 wpa-* 选项。可以在 usr/share/doc/wpasupplicant/README.Debian.gz 中查看示例和使用说明。最常见的选项是 wap-ssid（定义连接的无线网络名称）和 wpa-psk（定义了接入网络的口令或验证密钥）。

```
iface wlan0 inet dhcp
  wpa-ssid MyNetWork
  wpa-psk plaintextsecret
```

5.1.3 命令行工具systemd-networkd

ifupdown 作为 Debian 平台使用的传统工具，在很多服务器版或其他精简版系统中都是默认安装的。还有一个新工具也是非常值得推荐的：systemd-networkd。它与 systemd 集成在了一起，由此它成为了一个非常不错的选择。与 ifupdown 相反，它并不是专门基于 Debian 发行版开发的，它被设计得非常小巧、高效，并且如果你对 systemd 文件的语法有一定的了解，就会发现配置它也是相当容易的。如果你觉得 NetworkManager 体积臃肿又难于配置，那么它就是你的理想选择。

可以通过/etc/systemd/network/目录下的.network 文件来配置 systemd-networkd。或者，也可以在/lib/systemd/network/中找到它的本地配置文件，或在/run/systemd/network/中找到运行时生成的临时配置文件。这些文件的格式记录在 systemd.network（5）文件中。其中的 Match 部分指明了应用配置的接口。在这里可以定义接口的许多属性，包括网卡 MAC 地址或设备类型。在 Network 部分详细定义了具体的网络配置。

示例 5.1 /etc/systemd/network/80-dhcp.network 中的 DHGP 配置文件

```
[Match]
Name=en*

[Network]
DHCP=yes
```

示例 5.2 /etc/systemd/network/50-static.network 中的静态配置

```
[Match]
Name=enp2s0

[Network]
Address=192.168.0.15/24
Gateway=192.168.0.1
DNS=8.8.8.8
```

需要注意的是，system-networkd 在默认情况下是被禁用的，所以如果想要使用它，需要手动启用。它依赖于 systemd-resolved 服务，正确地完成 DNS 解析，也就是说需要把 systemd-resolved 管理的/run/system/resolve/resolv.conf 符号链接到 etc/resolv.conf。

```
# systemctl enable systemd-networkd
```

```
# systemctl enable systemd-resolved
# systemctl start systemd-networkd
# systemctl start systemd-resolved
# ln -sf /run/systemd/resolve/resolv.conf /etc/resolv.conf
```

尽管 systemd-networkd 存在一些限制，比如缺乏对无线网络的支持，但我们可以依赖已有的 wpa_supplicant 进行无线网络配置。然而，它在服务器、虚拟机或者那些最初就是为了管理网络而开发的系统中，就会显得特别有用。在你需要管理各种各样的虚拟网络设备时，systemd-networkd 能够减少管理不同网络环境的工作量。

5.2　管理UNIX用户和用户组

UNIX 用户和组的数据库由/etc/passwd（用户列表）文件、/etc/shadow（加密的用户密码）、/etc/group（组列表）和/etc/gshadow（加密的组密码）文件组成。它们的格式分别记录在 passwd(5)、shadow(5)、group(5)和 gshadow(5)中。虽然这些文件可以用 vipw 和 vigr 这样的工具手动编辑，但还是推荐大家使用一些更高级的工具，来执行用户或用户组修改设置操作。

5.2.1　创建用户账号

在使用 Kali 时我们经常用 root 直接登录，但由于各种原因，特别是当你使用 Kali 作为实体机操作系统时，还是需要创建非特权用户账号的。添加用户最典型的方式是使用 adduser 命令，它需要一个必需的参数：要创建新用户的用户名。在使用 adduser 命令创建账号时，命令会询问几个与该用户相关的问题。在它的配置文件/etc/adduser.conf 中包括许多不同的设置。例如，你可以定义可使用的用户标识符范围（uid），决定用户是否共享一个公共组，定义默认 shell 等。

创建一个账号时将会以/etc/skel 为模板，将模板目录下的所有文件复制到新添加用户的主目录下。它为用户提供了一组标准目录和配置文件。

在某些情况下，通过把一个用户添加到一个用户组（除了默认主组），以授予它额外的权限，这个办法在用户授权操作中非常有效。例如，如果一个用户在 sudo 组，那么可以通过 sudo 命令来使这个用户拥有完全的管理权限。可以使用 adduser user group 这样的命令，把用户加到一个组里面。

使用 getent 查看用户数据库

Getent（get entries）命令通过调用/etc/nsswitch.conf 文件配置的 name server switch（NSS）模块中合适的方法，来查看系统中的各种数据库（包括用户数据库和用户组数据库）。

这个命令需要一个或者两个参数：要去查询数据库的名称和一个可能需要的关键字。例如，我们可以使用 getent passwd kaliuser1 这个命令，返回用户数据库中关于 kaliuser1 的信息。

```
root@kali:~# getent passwd kaliuser1
kaliuser1:x:1001:1001:Kali User,4444,123-867-
     ➥ 5309,321-867-5309:/home/kaliuser1:/bin/bash
```

5.2.2 修改一个已存在的用户名及口令

以下命令可以用来修改存储在数据库中的用户信息。

- passwd：允许普通用户改变它们的密码，从而更新/etc/shadow 文件。
- chfn：（修改用户名）只针对超级用户（root），可以用来修改 GECOS 字段（用户信息，例如用户全名）或“general information”字段的内容。
- chsh：（修改 Shell）命令可以用来修改用户登录的 shell。只能在/etc/shells 列表中的 shell 之间进行切换，但是管理员不受这个限制，可以设定到任何 shell 上。
- chage：（修改有效期）命令允许管理员通过设置用户名作为参数，对该用户修改密码过期时间，或使用-l user 查看该用户的密码过期设置。或者，使用 passwd -e user 命令使该用户当前密码过期，这将迫使用户在下次登录时必须修改密码。

5.2.3 禁用一个账号

出于调查的目的，或者有长时间不登录的用户，你可能需要禁用某个账号（锁定用户）。被禁用的账号不能登录或访问主机，但账号信息会保留，也没有任何相关文件或数据被删除，仅仅是无法访问。可以通过使用 passwd -l（lock） user 命令来禁用一个账号。重新启用账号可以使用类似的方式，使用-u（unlock）参数来进行启用。

5.2.4 管理UNIX用户组

addgroup 和 delgroup 两个命令分别用来添加或删除一个用户组。groupmod 命令用

来修改用户组的信息（gid 或标识符）。命令 `gpasswdgroup` 更改组密码，而 `gpasswd -r` 命令可以删除组密码。

多个组的工作原理

一个用户可能属于多个组。一个用户的主群组在初始用户配置时就会被创建。默认情况下每个文件都属于它的创建者，也从属于这个用户的主群组。但这并不能满足所有情况。例如，当用户需要在由以其他群组为主群组中的用户共享的目录工作时，用户可以用使用 `newgrp` 命令开启一个新的 shell，或者使用 sg 命令临时切换到另一个可用组执行一次命令，这两种方式来可用来修改所属群组。这些命令也可以让一个用户加入当前不在的群组中。但如果这个群组设置了密码保护，那么在执行这些命令之前都需要提供正确的口令。

反过来，用户也可以修改这个文件所在目录创建时生成的 setgid 值，具体可以参考 setgid 目录和 sticky 值。

命令 `id` 可以用来显示当前用户状态，包括用户标识符（uid）、当前主群组（gid），还有它所属其他群组列表。

5.3　配置服务

在本节中，我们将介绍 Kali 系统中的服务（有时称为进程），或作为后台进程运行的程序和系统中的各种功能。我们首先来讨论配置文件，然后介绍一些重要服务（比如 SSH、PostgreSQL 和 Apache）的配置方法。

5.3.1　配置一个特定程序

当你想部署一个没安装过的软件包时，最好分若干阶段来进行。首先，你应该阅读这个软件包的安装说明。`/usr/share/doc/package/README.Debian` 文件是一个很好的例子。这个文件包含了关于软件包的很多信息，包括需要参考的其他文档。阅读这个文件，它往往会告诉你一些常见错误和常见问题的解决办法，那将会节省你很多时间，避免很多麻烦。

接下来，应该看看软件的官方文档。参考 6.1 节的内容可以知道如何找到各种文档的来源。`dpkg -L package` 命令可以列出这个软件包所包含的文件列表，可以通过这个列表快速定位到需要的文档（以及位于/etc/下的配置文件）。同时，`dpkg -s package` 命令将会显示包的源数据情况，并显示所有可能的依赖包或建议方案。在这里你可能会找到一些关于

此软件包的文档，从而帮助你更轻松地配置这个软件。

最后，查看配置文件。配置文件本身通常包含许多注释，详细介绍了各种可能需要的配置参数。但有的时候，你使用的软件运行时加载的配置文件里，就可能就只有那么一句没有任何注释的配置语句。还有一种可能，配置文件可能被存储在`/usr/share/doc/package/examples/`目录下，它们可能是一个最基本的配置文件。

5.3.2 配置SSH远程登录

SSH 服务允许你远程登录到一台机器，传输文件或执行命令。它提供了标准工具，用来连接远程机器的客户端（`ssh`）和服务端（`sshd`）。

openssh-server 在 kali 中默认是已经安装好的，但是 SSH 服务默认是被禁用的，因此引导时并没有启动。可以使用 `systemctl enable ssh` 命令配置在引导时启动 SSH 服务，或者使用 `systemctl start ssh` 命令动手启动它。

SSH 服务有一个相对完善的默认配置，但鉴于其强大功能和容易受到攻击的敏感性，我们最好还是知道一些能够对其配置文件`/etc/ssh/sshd_config` 做些什么。所有的选项都记录在 `sshd_config(5)`中。

默认情况下，是不允许 root 用户使用密码来登录的，需要先用 `SSH -keygen` 设置一个 SSH 密钥。你可以通过把 PasswordAuthentication 设置为 `no`，将这一安全属性扩展到所有用户，或者可以通过设置 PermitRootLogin 为 `yes`，解除这个安全限制（默认是 prohibit-password）。SSH 服务默认监听端口是 22，可以改变这个端口号。

调整好了新的设置后，需要输入 `systemctl reload ssh` 命令重载配置，以便使配置生效。

生成新的 ssh host key

每个 SSH 服务都将经过加密的私钥称为 SSH host key，并且把它们保存在`/etc/ssh/ssh_host_*`文件下。

以全盘复制的方式来安装系统时（不是通过 Debian 的安装镜像全新安装），会把原有的 SSH host key 也复制过来，原来的 root 用户与密码依然可以使用。可以使用下面的命令来重置它：

```
# Passwd
[...]
# rm /etc/ssh/ssh_host_*
# dpkg-reconfigure openssh-server
# service ssh restart
```

5.3.3　配置PostgreSQL数据库服务

PostgreSQL 是一个数据库服务。其很少单独使用，常被许多其他服务用来存储数据。

这些服务通常会使用网络访问数据库服务器并进行身份验证。因此，启用这些服务需要创建 PostgreSQL 数据库，并在数据库中配置有适当权限的用户账号。为了能够配置用户，我们必须要让数据库服务处于运行状态，可以使用 `systemctl start postgresql` 命令来启动。

各种版本的 PostgreSQL 数据库支持	PostgreSQL 允许各个版本的数据库协同安装使用。多个节点可以被同一个数据中心控制节点管理，形成一个数据库集群。实现集群管理的配置文件存储在`/etc/postgresql/version/cluster-name/`目录下。 为了能让集群内的多个数据库互不影响地正常工作，每一个节点都会有一个通信端口（通常是 5433）。`postgresql.service` 文件是一个空的 shell，利用它我们可以很容易把各个节点有效聚集到自己所在的集群。

连接方式和身份验证

默认情况下，PostgreSQL 侦听传入连接有两个路径：本地接口 TCP 协议的 5432 端口和基于文件的套接字`/var/run/postgresql/.s.PGSQL.5432`。这两个监听方式，可以通过调整 `postgresql.conf` 中的各个参数来进行配置：`listen_addresses` 代表监听的地址，`port` 参数设置 TCP 端口号，并可以用 `unix_socket_directories` 来定义创建套接字的目录。

这些参数综合起来决定了如何连接数据库，以及用哪种方法进行验证。`pg_hba.conf` 配置文件定义了允许谁、以何种身份验证、连接什么会话。默认情况下，在基于文件套接字的连接中使用 UNIX 用户账号作为 PosgreSQL 用户，如果没有特殊设定，就不需要进一步的验证了。在 TCP 连接中，PostgreSQL 要求用户通过用户名和密码进行身份验证（不是一个 UNIX 用户名/密码，而是一个由 PostgreSQL 本身管理的用户）。postgres 用户是专门管理数据库的，并拥有完全的管理权限。我们将使用这个用户来创建新用户和数据库。

创建用户和数据库

`createuser` 命令用来添加一个新用户，而 `dropuser` 命令则用来删除一个用户。同样，`createdb` 命令添加一个新的数据库，`dropdb` 命令删除一个数据库。每个命令都有自己的帮助手册，但是我们还是需要讨论一下其中的一些选项。每个命令通常作用于默认簇

（端口 5432），但可以使用`--port=port` 的命令，修改其他数据簇的用户和数据库的相关参数。

这些命令必须连接到 PostgreSQL 服务器，才能执行它们的工作，并且必须为经过身份验证的用户提供足够权限，才可以执行指定操作。实现这一点最简单的方法是使用 postgres UNIX 账号来通过套接字连接：

```
# su - postgres
$ createuser -P king_phisher
Enter password for new role:
Enter it again:
$ createdb -T template0 -E UTF-8 -O king_phisher king_phisher
$ exit
```

在上面的示例中，`-p` 参数使 `createuser` 创建新的 king_phisher 用户并直接为其设定密码。我们再看 `createdb` 命令，`-o` 参数用来指定新数据库的拥有者（拥有者有充分创建表和授予权限等的权限）。如果希望使用 Unicode 字符的编码类型，添加`-e UTF-8` 这一参数来设置编码格式，另外还需要使用`-t` 参数来为数据库选择模板。

现在可以测试了。我们可以使用 king_phisher 用户（`-U king_phisher`）通过套接字连接到数据库监听 localhost 的节点（`-h localhost`）。

```
# psql -h localhost -U king_phisher king_phisher
Password for user king_phisher:
psql (9.5.2)
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits:
     ➥ 256, compression: off)
Type "help" for help.

king_phisher=>
```

可以看到连接成功了。

管理PostgreSQL集群

首先，值得注意的是，“PostgreSQL 集群”的概念是一个在 Debian 系统中额外增加的概念，你在任何官方 PostgreSQL 文档中都找不到有关它的描述。从 PostgreSQL 工具的角度来说，节点就好像是一个数据库服务器运行在特定端口上的实例一样。

也就是说，Debian 系统的 `postgresql-common` 包提供了各种各样的工具，来管理这样的集群，例如，pg_createcluster、pg_dropcluster、pg_ctlcluster、pg_upgradecluster pg_renamecluster 及 pg_lsclusters。在这里我们不可能讲解所有这些工具，但是你可以参考它

们各自的帮助手册获得更多信息。

必须知道，当你在系统中安装了一个新版本的 PostgreSQL 时，将创建一个新节点，并运行在旧版本的下一个端口上（通常是 5433 端口）。除非你主动从旧的数据库集群迁移到新的集群，否则你使用的还是旧版本。

可以使用 `pg_lsclusters` 命令，来查看所有节点的列表，以及所有节点的状态。更重要的是，可以使用 `pg_upgradecluster old-version cluster-name` 命令，把你的集群自动迁移到最新版本的数据库中。为了能够成功迁移，首先需要清除新版本生成的一些集群列表（使用 `pg_dropcluster new-version cluster` 命令）。在此过程中旧集群并没有被丢弃，但它也不会自动启动。待确认迁移后的集群已经可以正常工作后，就可以把它丢弃掉了。

5.3.4　配置Apache

标准的 Kali Linux 系统默认已经安装了 *apache2* 软件包的 Apache Web Server 功能。作为一个网络服务，它在默认情况下是被禁用的。我们可以使用 `systemctl start apache2` 命令手动启用。

随着越来越多应用被部署成 Web 应用程序，了解用于运行 Web 网站的 Apache 相关知识变得越来越重要了。

Apache 是一个模块化的服务，它的许多功能都可以在初始化过程中通过加载外部模块来实现。默认配置只实现了最基本的功能，但是通过 `a2enmod module` 命令可以很容易启用模块。通过 `a2dismod module` 命令可以禁用模块。这些启用和禁用的命令实际上是在 /etc/apache2/mods-enabled 目录下创建（或删除）指向实际存储在/etc/apache2/mods-available / 文件的符号链接。

Apache 有许多可用的模块，但是有两个值得优先了解，那就是 PHP 和 SSL。用 PHP 编写的 Web 应用程序，可以被 Apache Web server 提供的专用模块 libapache-mod-php 包执行。

Apache 2.4 创造性地包含了 HTTP（HTTPS）安全所需要的 ssl 模块。它首先需要使用 `a2enmod ssl` 命令启用 `ssl`，然后必须将相应参数添加到配置文件中。`/etc/apache2/sites-available/default-ssl.conf` 提供了一个配置示例。在 `http://httpd.apache.org/docs/2.4/mod/mod_ssl.html` 网站上我们可以查询到更多信息。

标准 Apache 模块的完整列表可以在 `http://httpd.apache.org/docs/2.4/mod/index.html` 上找到。

在默认配置下，Web 服务器监听的端口是 80（在/etc/apache2/ports.conf 配置文件中），网站的默认目录在配置文件（/etc/apache2/sites-enabled/000-default.conf）中被设置为/var/www/html/。

配置虚拟主机

虚拟主机可以被看成是 Web 服务的额外身份。同一个 Apache 进程可以同时服务多个网站（比如 www.kali.org 和 www.offensive-security.com）。因为对网站的 HTTP 请求中嵌入了网站名称和 URL 本地部分（这个特性被称为基于名称的虚拟主机）。

默认配置的 Apache 2 启用了基于名称的虚拟主机功能。此外，有一个默认存在于 /etc/apache2/sites-enbled/000-default.conf 的虚拟节点配置文件。当没有找到匹配客户端发送的请求时，将会用到这个虚拟主机。

重要提示

这些没有指定虚拟主机的请求总是被第一个定义的虚拟节点处理，这就是为什么 Apache 要创建 000-default.conf 配置文件的原因，其用于处理那些没有设定如何处理的请求。

每个虚拟主机在/etc/apache2/sites-available/上都有关于它的描述性文档。该文件通常是主机对应网站地址的 conf 后缀文件（例如 www.example.com.conf）。可以使用 a2ensite www.example.com 命令来启用一个网站。在/srv/www.examplr.com/www/（在配置文件中 DocumentRoot 选项中定义了路径）目录下有一个已经写好的最简单的虚拟主机配置文件，可供大家参考：

```
<VirtualHost *: 80>
ServerName www.example.com
ServerAlias example.com
DocumentRoot /srv/www.example.com/www
</VirtualHost>
```

也可以考虑在配置 Apache 时添加 CustomLog 和 ErrorLog 功能，以便将日志输出到专用于虚拟主机的日志文件中。

常用指令

下面简要讨论一些常用的 Apache 配置指令。

主要的配置文件通常包括几个目录块；它们把操作对应到了相应的服务目录里。一个目录块通常包括 Options 部分和 AllowOverride 部分：

```
<Directory /var/www>
Options Includes FollowSymLinks
AllowOverride All
DirectoryIndex index.php index.html index.htm
</Directory>
```

`DirectoryIndex` 指令里的内容是当收到对目录块的访问请求时尝试去执行的一个文件列表。列表中按顺序第一个存在的文件将会被作为响应发送。

在 `Options` 指令后面可以添加可选配置选项。`None` 代表禁用所有的功能；相应地，`All` 参数将会启用除 `MultiViews` 之外的所有功能。

可用的选项包括：

- `ExecCGI`——允许执行 CGI 脚本。
- `FollowSymLinks`——告诉服务器允许对符号链接进行跟进操作，并且返回的相应数据应该包含此链接的内容。
- `SymLinkslfOwnerMatch`——如果一个符号链接的源和目标同属于一个拥有者，则允许跟进符号链接。
- `Includes`——启用服务端包含服务（SSI）。确保那些嵌入页面中的指令能够正确执行和响应。
- `Indexes`——告诉服务器如果在某个目录中不包含 `index.html` 页面（由 `Directorylndex` 指定），则显示目录内容的列表。
- `MultiViews`——提供内容协商机制，该选项默认被禁用。其可以使服务器返回一个与浏览器中配置语言相匹配的页面。

需要身份验证

在某些情况下，网站访问的授权是非常重要的，只有提供了正确的用户名和密码才被授予访问权限。

`.htaccess` 文件包含当有请求涉及存储该文件中的目录元素时，要去执行的 Apache 配置指令。这些指令是递归的，并可将范围扩展到所有的子目录。

大多数可以在目录块中出现的指令在`.htaccess` 文件中也是合法的。`AllowOverride`

指令列出了所有可以通过.htaccess 启用或禁用的选项。此选项的一个常见用法是用来限制 ExecCGI，以便让管理员选择允许哪些用户在通过 Web 服务器身份验证（www-data 用户）后运行程序。

示例 5.3　htaccess 文件需要身份验证

```
Require valid-user
AuthName "Private directory"
AuthType Basic
AuthUserFile /etc/apache2/authfiles/htpasswd-private
```

基本的身份验证机制并不安全　在上面的例子中，最基本的身份验证过程中的密码是被设置成明文保存的（只是采用了最基本的 base64 对密码进行重新编码，而不是采用一种安全的加密算法来进行加密）。这种存储方式在互联网上随处可见。如果你很重视安全，所有的 HTTP 会话都应该通过 TLS 加密。

/etc/apache2/authfiles/htpasswd-private 文件保存了一个存有用户名与密码的列表，通常可以使用 htpasswd 命令来操作。例如，下面的命令用于添加用户或修改用户密码：

```
# htpasswd /etc/apache2/authfiles/htpasswd-private user
New password:
Re-type new password:
Adding password for user user
```

访问限制

Require 指令控制目录（及其逐层递归的子目录）的访问限制。

它可以基于许多标准来做出访问控制，可以基于客户机的 IP 地址来设置访问控制限制，还可以把几个限制策略组合在一个 RequireAll 块来进行限制。

例如，可以使用以下指令来限制只能通过本地网络进行访问：

```
Require ip 192.168.0.0/16
```

5.4　服务管理

Kali 使用 systemd 作为其初始化系统、引导服务启动顺序的全功能服务管理器，负责启动和监控各个服务。

可以使用 systemctl 命令来查询和控制 systemd。在没有输入任何参数时，等同于执

行 `systemctl list-units` 命令，输出一个活动服务的列表。如果运行 `systemctl-status` 命令，将会输出一个运行服务的分层概述。比较两种输出结果，立刻就能看到有多种类型的元素，而服务只是其中一种。

每个 *service* 单元代表一个服务，服务的描述文件通常存储在`/lib/systemd/system/`、`/run/systemd/system/`或者`/etc/systemd/system/`目录下；根据优先级顺序来进行排列。这些文件可能被来自同目录下其他服务的 `service-name.service.d/*.conf` 文件修改。这些单元文件被记录成文本文件的格式，这个灵感来自于著名的 Microsoft Windows ini 文件，在不同`[section]`之间用 `key = value` 对描述配置。

在这里，我们看一个`/lib/systemd/system/ssh.service` 的服务文件：

```
[Unit]
Description=OpenBSD Secure Shell server
After=network.target auditd.service
ConditionPathExists=!/etc/ssh/sshd_not_to_be_run

[Service]
EnvironmentFile=-/etc/default/ssh
ExecStart=/usr/sbin/sshd -D $SSHD_OPTS
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure
RestartPreventExitStatus=255
Type=notify

[Install]
WantedBy=multi-user.target
Alias=sshd.service
```

Target 单元是 `systemd` 设计的另一个组成部分。它们表示你希望活动单元（即正在运行服务）达到的理想状态。它们主要作为将其他单元依赖分组的形式存在。当系统启动时，它将使得所需单元达到 `default.target`（这是一个到 `graphical.target` 的符号链接，而这又依赖于 `multi-user.target`）状态，以此来保证这些 target 的所有依赖关系都会在启动时被激活。

这种依赖关系是通过目标单元的 `Wants` 指令来指定的。但我们不需要添加编辑新的依赖关系，我们只需要创建一个指向`/etc/systemd/system/target-name.target.wants/`目录依赖单元的符号链接就行了。这正是 systemctl 启用 foo.service 服务时做的事情。当你要启用一个服务时，就告诉 systemd 对服务单元文件`[Install]`部分 WantedBy 条目中列出的目标

添加一个依赖关系。相反，如果 systemctl 要去禁用 foo.service，只要删除相同符号链接和相关依赖就可以了。

enable 和 disable 命令不会改变任何有关服务的当前状态。它们只会在下一次启动引导时产生变化。如果你想立即运行某个服务，应该执行 systemctl start foo.service 命令。相反，你可以使用 systemctl stop foo.service 命令来停止它。你还可以使用 systemctl status foo.service 命令检查服务的当前状态，这条命令通常会包含关联日志的最新几行。当改变一个服务的配置后，你可能希望重新加载或重新启动它，操作的命令分别是 systemctl reload foo.service 和 systemctl restart foo.service。

```
# systemctl status postgresql
● postgresql.service - PostgreSQL RDBMS
   Loaded: loaded (/lib/systemd/system/postgresql.service; disabled; vendor
      ➥ preset: disabled)
   Active: inactive (dead)
# ls -al /etc/systemd/system/multi-user.target.wants/postgresql.service
ls: cannot access '/etc/systemd/system/multi-
    ➥ user.target.wants/postgresql.service': No such file or directory
# systemctl enable postgresql
[...]
# ls -al /etc/systemd/system/multi-user.target.wants/postgresql.service
lrwxrwxrwx 1 root root 38 Apr 21 16:21 /etc/systemd/system/multi-
     ➥ user.target.wants/postgresql.service ->
     ➥ /lib/systemd/system/postgresql.service
# systemctl status postgresql
● postgresql.service - PostgreSQL RDBMS
   Loaded: loaded (/lib/systemd/system/postgresql.service; enabled; vendor
      ➥ preset: disabled)
   Active: inactive (dead)
# systemctl start postgresql
# systemctl status postgresql
● postgresql.service - PostgreSQL RDBMS
   Loaded: loaded (/lib/systemd/system/postgresql.service; enabled; vendor
      ➥ preset: disabled)
   Active: active (exited) since Thu 2016-04-21 16:22:29 EDT; 2s ago
  Process: 6355 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
 Main PID: 6355 (code=exited, status=0/SUCCESS)

Apr 21 16:22:29 kali-rolling systemd[1]: Starting PostgreSQL RDBMS...
Apr 21 16:22:29 kali-rolling systemd[1]: Started PostgreSQL RDBMS.
```

5.5　小结

在这一章中，我们了解了如何配置 Kali Linux。介绍了如何配置网络，谈到了用户和用户组，并讨论了如何创建和修改用户账号、设置密码、禁用账号和管理组。最后，我们讨论了服务，并解释了如何设置并维护常用服务，特别是 SSH、PostgreSQL 和 Apache。

要点提示：

- 在一个标准的桌面安装中，已经安装了 NetworkManager，并且可以通过右上角的菜单和安装好的 GNOME 控制中心对它进行管理和配置。
- 可以在命令行输入 `ifup` 和 `ifdown` 命令来配置你的网络，它们的使用说明在 `/etc/network/interfaces` 配置文件里。*systemd* 初始化系统下的 *systemd-networkd* 是一个不错的新工具。
- 默认情况下，UNIX 用户和组的数据库由 `/etc/passwd`（用户列表）、`/etc/shadow`（用户加密后的密码）、`/etc/group`（用户组列表）、`/etc/gshadow`（组加密密码）几个文本文件联合组成。
- 可以使用 `getent` 命令查看用户数据库和其他系统数据库。
- `adduser` 命令在创建账号之前会问几个问题，但这是最直接的创建一个新用户的方法。
- 好几个命令都可用于修改用户数据库中的特定字段，包括 `passwd`（更改密码）、`chfn`（修改用户名和其他描述信息）、`chsh`（改变登录 shell）、`chage`（更改密码过期时间），以及 `passwd -e user`（用户下次登录时强迫用户修改密码）。
- 一个用户可以是一个或多个组的成员。可以使用下面的命令来修改用户组的标识值：`newgrp`，修改当前组的 ID；`sg`，临时切换到另一个组执行。每一个目录都有自己的 `setgid` 标志位值，在创建目录的时候其自动属于创建用户的所在组，此外，`id` 命令用来显示用户的当前状态，包括该用户所在组群的列表。
- 可以通过 `systemctl start ssh` 命令来手动启动 SSH，或是通过 `systemctl enable ssh` 命令永久开启 SSH 服务。默认配置 root 用户禁止使用密码登录，必须先用 `SSH -keygen` 命令来设置 SSH 密钥。
- PostgreSQL 是一种数据库服务。它很少被单独使用，却经常被其他服务用来存储数据。
- 一个标准的 Kali Linux 安装，应该已经通过 `apache2` 包安装好了 Apache Web 服务。作为一个网络服务，它在默认情况下是被禁用的。可以通过 `systemctl start apache2` 命令来启动它。

在默认的配置中，Apache 的侦听端口是 80（`/etc/apache2/ports.conf`），服务网

页默认（根据`/etc/apache2/sites-enabled/000-default.conf` 中的配置）存储在`/var/www/html` 目录下。

现在我们已经了解了 Linux 的基本原理，以及 Kali Linux 的安装和配置，下一章我们将讨论如何排除在使用 Kali 时遇到的故障，以及在遇到困难时使用什么工具和技巧去解决。

练习题

练习1——配置管理Kali系统中的用户

1．创建一个标准的用户账号。把这个新账号添加到管理员组中。

练习2——配置网络

1．停止 Network Manager 服务，并在引导启动项中把它设置为禁用。

2．把 Kali 系统中的 eth0 网卡配置为 DHCP 获取网络配置。

3．关闭掉 eth0 接口。

4．通过配置相应的`/etc/network/interfaces` 文件，使无线 USB 网络适配器连接到无线网络。

练习3——配置服务第1部分

1．配置 SSH 服务使其允许使用密码登录 root 账户（提示：PermitrootLogin）。

2．启动 SSH 服务并以 root 用户进行登录。

3．将 SSH 服务配置为引导时启动。

4．更改 root 密码并生成新的 SSH 主机密钥。

5．安装 hostapd 并把它配置为引导时启动，使你的 Kali 实例成为无线网络的接入点。并配置此服务来完成无线接入点的构建操作！

练习4——配置服务第2部分

在这个练习中，我们将安装 masscan。它是一个很棒的工具，整个安装过程非常有助于我

们回顾在本章中探讨的配置概念。整个过程分为以下几个步骤：

1．安装并启动 Apache 和 PostgreSQL 服务。

2．将 Apache 和 PostgreSQL 配置为引导时启动。

3．结合 Apache 和 PostgreSQL，安装 masscan 并构建其安全扫描的 Web 接口。

4．导入扫描记录并查看结果。

5．利用 htaccess 对 Apache 进行访问控制保护。

进阶练习

练习5——树莓派接入点

如果你还没有树莓派 3，你真的应该去买一个。它真的太棒了，而且也很便宜。在本练习中，我们将配置 Raspberry Pi 3 构建一个无线热点，接入的用户可以通过它访问 Internet。此练习非常棒，因为你需要为 Raspberry Pi 安装一个 Kali 操作系统，并且编辑文件，更改文件权限，配置网络接口，安装和配置服务，配置 iptables 规则等。可以说这是一个非常全面的练习。

以下是你需要做的事情：

1．在 Raspberry Pi 3 上安装 Kali 。你可以自由选择镜像，但是如果你这样做，你可能会遇到更多需要处理的麻烦问题。如果你不确定，请选择一个有完整树莓派安装解决方案的系统镜像。

2．在 AP 上设置 WPA2 安全属性。

3．将 eth0 配置为 DHCP，并将 wlan0 配置为静态的。

4．为 Raspberry Pi 配置一个为无线接入终端分配 IP 的 DHCP 服务，并为其分配一个具有 12 小时租约功能的 DHCP 池。

5．设置 SSH 服务为在引导时启动，以便你可以在开机后立即连接到 Raspberry Pi。

6．将所有出站流量（包括 DNS）从 wlan0 转发到 eth0。

7．允许从 eth0 到 wlan0 的入站建立（stateful）连接。

8．提示：虽然你还没有了解 hostapd 或 dnsmasq，但你需要在本练习中使用它们。

9．特别提示：虽然这个练习不是专为 Kali 设计的，但这个练习非常有价值，值得去体会并进行扩展。

第6章　自助解决问题并获得帮助

内容

- 文档资源
- Kali Linux 社区
- 提交一个友好的 bug 报告
- 小结

不管你多有经验，你都无法避免地会遇到问题。为了解决一个问题，你得理解这个问题，然后利用各种资源找到一个解决方案。

在这一章中，我们来聊聊各种好用的信息源，以及在遇到问题后查找解决方案的一些好用的方法。我们将去看看 Kali Linux 的社区，了解相关的网络论坛以及网络聊天（IRC）频道。最后，讨论 bug 的报告机制和如何利用 bug 跟踪系统解决问题，并介绍一些有助于你完成自己 bug 报告的策略和方法，以便让未归档的问题，能够得到快速有效的解决。

6.1　文档资源

在你能搞懂引发问题的根源之前，需要从原理上明白每个程序所扮演的角色。最好的方法是去研究程序的各个文档。但这些文档往往数量巨大且很分散，因此也需要知道如何找到这些文档。

如何避免 RTFM 回答	RTFM 是“read the fucking Manual”的首字母缩写，意思是“读那该死的文档”，但是也可以把其理解成一个文雅点的缩写“read the fine manual”，这个短语经常被用来回复新手的提问，表达略带厌烦的情绪。有人认为这种回应略显粗鲁，毕竟对于提问的人来说，他们未必会有耐心去阅读文档。但也有人认为，这种经典的回应方式聊胜于无（至少指出已有的文档包含了相关的信息），毕竟也比直接得到一个满是抱怨并且十分冗长的答复强。

在大多数情况下，如果有人用“RTFM”来回应你的提问，通常并没有什么恶意。你在网络上发布问题时，尽可能把在发布问题之前你所尝试的处理告诉大家。告诉大家你找到的相关资料，以及你自己找到线索的步骤和方法。虽然会花一些时间，但这会让大家知道你并不是一个懒人，而是一个对知识非常渴望的人。Eric Raymond 曾发表过一篇指引性的文章，该文章就如何提问给出了建议，这能有效避免常识性的错误并得到有用的答案。文章地址是：

➨ `http://catb.org/~esr/faqs/smart-questions.html`

6.1.1　手册页

手册页（manual pages）通常都是文笔精炼并包含了大量有用信息的文档。通过查找手册页，能快速知道各种命令的使用方法。只要简单地输入命令 `man manual-page`，就能显示所查命令的手册页。例如，你想了解 `cp` 这个命令的所有选项，可直接在 shell 控制台输入 `man cp`。

手册页不仅在我们使用命令行处理文件时可以提供帮助，其也可以在配置文件、系统调用及调用 C 语言库函数等方面提供支持。当遇到拥有相同名字的不同类型程序时，就可能造成冲突。例如，shell 有一个命令 `read` 和系统调用函数 `read` 重名。为此，手册页通过添加数字参数的方式，对不同种类的文档进行分类：

1. 可以通过命令行执行的命令
2. 系统调用（内核提供的系统函数）
3. 库函数（系统提供的函数库）
4. 设备（在类 UNIX 系统中，这些都是特殊文件，通常放置在/dev/目录下）
5. 配置文件（格式和规范）
6. 游戏
7. 宏命令和标准设置
8. 系统管理命令
9. 内核例程

你可以在使用手册页查看文档的时候指定对应的分类数字（section），例如浏览系统调用 read 的文档，可以键入 `man 2 read`。如果不指定具体的分类数字（section），系统会从小到大获取第一个数字作为默认值。因此直接输入 `man shadow` 时，系统就会返回 `shadow(5)`的文档内容，这是因为在分类页面 1～4 都没有 `shadow` 这个命令的手册页。

当然，如果你连命令的名字都不知道，手册页恐怕也无能为力了。遇到这种情况，我们可以用命令 `apropos` 来试试，这个命令能帮你查询到手册页（或者手册页中的简述内容）中符合你搜索关键字的内容。`apropos` 命令会返回在手册页中查找到的包含关键字的有关内容列表。如果关键字选择得恰当，你就能通过此工具很快得到想要的结果。

示例 6.1　使用 apropos 命令找到 cp 命令

```
$ apropos "copy file"
cp (1)               - copy files and directories
cpio (1)             - copy files to and from archives
gvfs-copy (1)        - Copy files
gvfs-move (1)        - Copy files
hcopy (1)            - copy files from or to an HFS volume
install (1)          - copy files and set attributes
ntfscp (8)           - copy file to an NTFS volume.
```

通过前向链接浏览文档　很多手册页通常在文档结尾处有一个“see Also”选项，里面会提到很多与查询内容相关或者相似的命令，或是一个外部的文档。如果最初显示的内容不够理想，你还可以通过这个线索找到你想要的结果。

除了 `man` 命令之外，还可以使用 `konqueror`（在 KDE 环境下）和 `yelp`（在 GNOME 环境下）命令来搜索手册页。

6.1.2　Info文档

GNU 项目为其大部分的程序编写了 info 格式的文档，这就是为什么很多手册页直接引用了相应的 info 文档内容的原因。这种格式有一定的优势，但默认的查看程序（也就是 `info` 命令）却会使得查看这些文档变得不那么直观。建议你最好安装一个 `pinfo` 工具。安装它时先输入命令 `apt update`，然后输入 `apt install pinfo` 命令，就可以完成安装（参考 8.2.2 节的内容）。

info 文档有一个层次结构，如果在调用 `pinfo` 命令时没有输入任何参数，它将会列出第一层所有有效的节点。通常来说，节点名称与命令名称相对应。

`pinfo` 命令可以使用键盘的上下箭头，很轻易地在节点之间进行导航。另外，也可以使用图形界面浏览器（更加友好的视图界面）来查看文档，例如 konqueror 或者 yelp。

如果考虑语言翻译，info 系统都是英文书写的，并没有翻译版本，这点与 man 系统不大一样。然而，当你尝试使用 `pinfo` 命令访问一个并不存在的 info 页面时，将会回退到同名的

man 页面（如果该页面存在），这时的页面有可能是被翻译了的。

6.1.3　软件包专用文档

每一个安装包都有自己的文档库。即使是最缺乏文档的软件包，也至少会包含 README 文件，该文件记录了一些主要的或者非常重要的信息。这些文档通常会被安装在`/usr/share/doc/package/`目录（这里的 package 代表软件包的名字）下。如果这个文档库特别大，它可能不会包含在这个程序的主软件包当中，而是可能被独立打包成一个叫作 package-doc 的文档包。在主安装包中常常会推荐或提到这个文档包，以便你能更容易地找到它。

在`/usr/share/doc/package/`目录当中同样有一些由 Debian 软件包维护者补充的文档，与传统的软件安装文档不同的是，在这些文档中增加了一些关于软件包的特性或完善的内容。README.Debian 文件指出了所有为适应 Debian 的软件包发布政策而发生修改的相关内容。changelog.Debian.gz 这个文件则按照时间的顺序描述了同一功能若干次改动间的区别，这对理解不同版本程序的演变过程很有帮助。最后，NEWS.Debian.gz 这个文件记录了管理员可能非常关心的一些功能变化。

6.1.4　网站

在大部分情况下，可以找到一个用于分发自由软件（free software）并通过某种方式将相关开发人员和用户聚集在一起的网站。这些网站通常是由以下一些信息以不同的方式组建起来的：官方文档、FAQ（常见问题列表）、邮件列表和归档内容等。通常，你所遇到的问题已经被很多人遇到并且上报过了，你很可能通过 FAQ、邮件列表归档等方式，就能找到解决的方法。即便没有找到，你还可以通过搜索引擎来寻找对你有帮助的资料，在你使用互联网搜索信息的同时，你检索问题的能力也得到了极大的锻炼。如果搜索引擎所返回的页面过多或者结果并非你所想要的，你可以添加 kali 或者 debian 等关键字，来限制显示的结果和相关的信息。

从错误到解决方案　如果软件返回了一个特定的错误信息，你可以将这段信息以半角双引号“”括起来（这样能保证以完整短语进行查找，而不是单个单词），输入搜索引擎中进行查找。在大部分情况下，第一条链接就会包括你所需要的答案。

在一些情况下，会出现一种经常遇到的错误信息，例如“Permission denied”。在

这种情况下，最好检查一下相关元素的权限情况如何（文件、用户 ID、组别等）。总之，不要总是依赖使用搜索引擎来找到解决问题的方法。否则，你会发现已经忘记使用常识去解决问题了。

如果你不知道这个软件网站地址，有很多种方法去获知。首先，检查软件包的元信息中的 Homepage 字段（命令为 `apt-cache show` 软件包名称）。另外，软件包的描述可能也包含了程序的官方网址。如果此处没有网址，可以查看`/usr/share/doc/package/copyright` 文件里有没有一个 URL 地址。你还可以使用搜索引擎（如 Google、DuckDuckGo、Yahoo 等）来查询这个软件的相关网站。

6.1.5 docs.kali.org上的Kali 使用文档

Kali 这个项目在 `http://docs.kali.org` 上维护了一个包含很多有用文档的网站。这些文档覆盖了大量使用 Kali Linux 时应该知道的内容，其中的很多操作和使用技巧（很像 how-tos）都是非常实用的。

➡ `http://docs.kali.org`

让我们来看一下这个网站涵盖的各个主题。

- 入门：为 Kali 新手提供的一系列说明，包括下载说明。
- Kali Linux Live：介绍如何使用最新 Kali Linux 系统的文档。
- 安装 Kali Linux：描述 Kali Linux 安装的各种文档，包括如何与其他操作系统并行安装。
- ARM 上的 Kali Linux：许多有关在基于各种 ARM 设备上运行 Kali Linux 的方法。
- 使用 Kali Linux：涵盖许多常见命令请求方法。
- 定制 Kali Linux：根据自己需求定制 Kali 系统所需要的补丁程序说明。
- Kali 社区支持：指向各个社区，你可以在这里获得支持，并了解如何提交 bug 报告。
- Kali Linux 策略：解释与其他 Linux 发行版相比，Kali Linux 的特殊之处。
- Kali Linux Dojo：Black Hat 和 DEFCON 会议的视频。

6.2 Kali Linux社区

全世界有很多不同的交流平台（例如论坛和社交网络），大家在这里讨论和研究 Kali Linux。这一节我们只介绍两个最主流的 Kali Linux 社区。

6.2.1　forums.kali.org上的Web论坛

Kali Linux官方的社区论坛的网址是forums.kali.org[1]。像每个论坛一样，你必须创建一个账号，以便你能够把你所遇到的问题上传到上面，从而在社区跟大家讨论你的问题。

在上传问题之前你应该先读一下这个论坛的规则：

➨ `http://docs.kali.org/commission/kali-linux-community-forums`

并不需要把这些都背下来，但是我们要知道在这里不允许讨论违法的事情，例如如何黑进其他人网络之类的话题。必须尊重社区的其他成员，要营造一个友好的社区氛围。广告和与话题无关的内容也尽力避免在社区里讨论。这个社区有足够多的分类，你可以在其中讨论关于 Kali Liunx 的各种内容。

6.2.2　kali-linux IRC的freenode即时聊天频道

IRC是一个即时聊天系统。通常这个聊天室被称为频道，大家在这个频道里，都围绕着一个有针对性的话题进行讨论和沟通。Kali Linux项目把这里的kali-linux频道放在了Freenode[2]的网络上。（可以把`chat.freenode.net`当作一个即时聊天服务器，在 6667 端口上进行加密聊天，在 6666 端口进行明文通信）。

如果你想使用IRC进行讨论，你可以使用一个IRC客户端，例如hexchat（有视图画面），或者irssi（命令行版本），或者webchat.freenode.net[3]基于浏览器的聊天客户端，都可以帮你加入讨论。

虽然加入这个讨论频道是很简单的事情，但是你仍然要注意，IRC 频道有它们自己的规矩，并且那里有一些频道管理员（他们的名字有一个@的前缀）。他们能强制执行一些规则：他们可以把你踢出这个频道（或者如果你不断触犯这个频道的规则，他们可以禁止你进入这个频道）。kali-linux 这个频道也不例外，它的规则记录在 `http://docs.kali.org/community/kali-linux-irc-channel` 上。

它的规则是：你必须是友好的，宽容的，并且通情达理。你应该尽量避免讨论一些离题的事情。特别是讨论那些违法的事情，破解软件、入侵、盗版软件、政治、宗教等问题都是被禁止讨论的。你的 IP 可是能被别人看到的！

1 http://forums.kali.org
2 http://www.freenode.net
3 http://webchat.freenode.net

如果你想要寻求别人的帮助，可以照着“如何来避免RTFM回答”中列出的建议去做。首先自己试着去研究，然后把你的研究结果告诉大家。当你需要追问的时候，请准确地描述你的需求（如果你有必要提供一段很长的描述，那么别把它直接贴到这个频道里，而最好是用一个称为Pastebin[1]的服务来代替，然后把这个Pastebin URL发到频道里）。

不要期望马上就能得到答案。尽管 IRC 是一个即时通信的平台，但是鉴于参与者来自于世界各地，所以时区和工作安排有较大差异。可能需要几分钟甚至几小时，才有人来响应你的问题。尽管如此，当有人@你的名字，或者回应你的话题时，聊天窗口会亮起并且大多数的 IRC 客户端都会通知你，所以保持你的客户端在线并且有点耐心。

6.3 编写一个友好的bug报告

如果你想尽办法还是没能搞定一个问题，那么其有可能是一个系统的 bug。在这种情况下，这个问题可能在一个已提交的 bug 报告中有记录。你可以去搜索一下 bug 报告看看能不能找到你的问题的解决办法。我们先来看看把一份 bug 报告发给 Kali、Debian 或者上游的开发者的步骤，以及如何根据你的需要提交你自己的报告。

bug 报告的要点是提供足够的信息，以便开发人员或维护者能够在处理问题程序时可以重现这个问题，调试其功能，并研发出一个修复补丁。这意味着你的 bug 报告必须包含适当的信息，必须提供给正确的人或项目团队。报告还必须有良好的书写和完整的描述，以便能够更快得到回复。

具体 bug 报告的提交过程取决于你将提交报告的平台(Kali、Debian、上游开发者)，但也有一些在各种情况下都通用的建议。下面，我们将讨论这些建议。

6.3.1 一般性建议

这里我们提供一些一般性的通用建议和指导意见，帮助你把 bug 报告描述得清楚、全面，以提高解决问题的可能性，也能使开发人员更及时地给予反馈。

如何沟通

用英文来写报告

自由软件社区都是国际化的，除非你认识谈话对象，否则你应该使用标准的英语来写报

1 http://pastebin.com

告。如果你的母语是英语，请使用简单的句子，避免结构过于复杂，给英语能力有限的人带来困扰。尽管大多数开发人员都非常聪明，但并不是所有人都有良好的英语语言能力。最好永远不要假设。

请尊重开发人员的工作

请记住大多数的自由软件开发者（包括这些 Kali Linux 的开发人员）都是非常友善的，并且他们把自己有限的业余时间投入这些软件的开发工作中，这才能够让大家免费使用。他们中的很多人都是志愿者。因此当你填写一个 bug 报告的时候，应该秉着尊敬的态度（即便这个问题很明显是开发者的错误），并且不要认为他们亏欠你什么。而是感谢他们所作的贡献。

如果你知道如何修改或者重新编译这个软件，也请你帮助这些开发人员对他们给你的补丁进行测试。这将会告诉开发者你也愿意贡献时间来帮助他们。

关注提交问题后的反馈，并且做好准备提供更多信息

在很多情况下，开发者会带着一些需求回来问你要更多信息，或者需要你帮忙使用不同配置或者使用另一个安装包试着重现这个问题。你应该尽可能快地给予回应。你回应得越快，你的问题被解决的可能性也就越大，因为这样会使你的问题在他们的脑海中被反复思考。

当你给予了及时的响应后，你还要注意别做得太过匆忙：提交数据一定要正确，并且包括了所有开发者要求的内容。否则他们不得不反复要同一个相关文件，这会非常影响处理效率。

bug报告都应该写些什么

介绍如何重现这个问题

为了能够重现这个问题，开发者需要知道你是怎么做的，在哪里发现的问题，还有你是如何安装的。

你应该提供清晰的步骤介绍，来描述如何重现这个问题。如果你需要用一些数据才能重现这个问题，那么请把相关文件附在这个报告上。尽量用最精简的指令来重现问题。

介绍问题出现的背景和你想要做的操作

解释你都做了哪些事情，以及你希望程序表现出什么现象。

在很多时候，只是因为你以一种没有被软件设计者想到的方式来使用软件才触发了 bug。解释一下你在想做什么操作时触发了这个问题，这将会让开发者更加清晰地了解问题是什么时候发生的。

还有一些情况，你描述为 bug 的行为实际上只是一个正常的行为。需要明确你预期程序会做些什么。这能让开发者明白应用环境是怎样的。这可以让他们改善这个操作，或者完善他们的使用说明，至少他们知道他们设计的程序困扰了一些使用者。

一定要详细具体

详细列出你使用软件的版本号，最好这个版本是他们目前支持的。当你提到一些你下载的东西时一定要附上完整的 URL。当你遇到一个报错信息时，把你看到的内容准确地引用下来。如果可能，最好附上屏幕输出消息或者截屏。附上相关的日志文件之前，首先确保隐去所有隐私的信息。

提及可能修正或者变通的方法

在你写报告之前，你可能自己尝试着去解决这个问题。讲述一下你都做了哪些操作并且结果如何。清楚标出，哪些是你实际操作的内容，还有哪些是你设想的部分。

如果你在网上找到了一些解决类似问题的方法，你可以把它们也附在报告里面，特别是你在 Debian 报告跟踪站或者上游 bug 报告网站中找到了一些类似的问题报告。

如果你找到了一个操作方法能够操作成功，请把它们详细记录下来。这将在其他的使用者遇到这个问题时帮助到他们。

详尽的 bug 报告总是好的

一两行的错误报告大都是不完善的，提供所有必要的信息通常需要至少几个段落（有的时候甚至是几页）。

尽可能提供所有你能提供的信息。尝试把所有相关的信息都写上，如果你不确定要提供多少信息，也要秉承一条，过多也好过没有。

如果你的报告实在太长了，那么请花点时间调整一下语言结构，并且在开始处写一个短一点的概述。

一些提示

避免填写重复的报告

在自由软件的世界里，所有 bug 跟踪过程都是公开的。问题报告都是公开可被浏览的，并且都是可以被搜索的。因此在你提交一个 bug 报告之前，请先确认有没有别人已经提交了相同的问题。

如果你找到了一个已经被别人提交的 bug 报告，你可以订阅这个贴子，并且对这个问题进行一些补充描述。别在留言处发表一些我也这样或者+的回复，不会有人理你的。但是你可以补充讲述更深入的内容，或者第一个提交者没有提到的内容，或者你的研究成果。

如果你没有找到任何关于你问题的报告，那么继续完成你的报告吧。如果你找到了相关资料，一定记得把它们写进去。

确保你用的是最新版本

对于开发者来说，当他们收到一个他们已经解决了问题的错误报告，或者他们无法在他

们使用版本（开发者通常都使用最新版本的软件）中重现的问题时，他们都会很烦躁。即便开发者依然对旧版本维持支持，他们通常也只提供安全更新和重大问题更新。请确定你的问题是不是存在版本过旧的问题。

因此你在填写一个 bug 报告之前，要确认你使用的是最新版本系统或者软件，并且你能够在一个场景下重现这个问题。

如果 Kali Linux 没有提供这个软件的最新版（kali-rolling 和 kali-bleeding-edge 都没有提供更新，可以参照 8.1.3 节的介绍），有以下两种解决办法：你可以尝试去第三方渠道获取并且手动更新它，或者你可以查看上游变更日志（或者看看 git commit 历史），如果这些地方都没有修正你的这个问题，那么你就不用去尝试更新了，直接准备提交 bug 报告吧。

不要在一个问题报告中提交多个问题

一个报告提交一个问题。这是因为，下面的讨论区会因为讨论多个问题而变得很乱，你可以针对每一个问题做出修复计划。如果你不这样做，对于一个 bug 却需要反复多次地确认，直到所有问题都解决才会关闭这个讨论，或者开发者不得不写一个本来你在第一步就应该写的补充报告。

6.3.2　在哪里提交问题报告

为了能够判断把问题报告提交给谁，你必须对这个问题有个良好的认识。并且必须判断出是程序的哪段代码出了问题。

在理想的情况下，你可以跟踪这个问题到系统上的文件，然后用 `dpkg` 命令去找出拥有这个文件的安装包，确定这个包的来源是哪里。例如，你在一个图像应用程序中发现了一个 bug。你在查看了运行进程目录（`ps auxf` 命令的输出结果）后，发现这个应用程序是通过 `/usr/bin/sparta` 从命令行启动的：

```
$ dpkg -S /usr/bin/Sparta
sparta: /usr/bin/sparta
$ dpkg -s sparta | grep ^Version:
Version: 1.0.1+git20150729-0kali1
```

你能看到 `usr/bin/sparta` 是由 sparta 安装包提供的，它在 1.0.1+git 2015-729-0kali1 中。由这个 version 参数，可以知道这个安装包来源的 Kali Linux 版本（或者是 Kali Linux 定义的版本）。如果那个安装包没有 kali 的标记，那么可能直接来源于 Debian 的基础软件包。

在提交关于 Debian 的 bug 之前一定要再次确认一下	如果你找到了一个属于 Debian 的 bug，你应该认真地报告给 Debian 这边，并由他们来进行修复。尽管如此，在做这些之前，请确认这个问题可以在 Debian 系统上得到重现，因为很有可能是因为 Kali 安装了其他安装包或者依赖程序造成的。 最简单的验证方式，就是新建一个 Debian 虚拟机来测试一下。你可以在 Debian 安装网站上找到一个安装镜像来对 Debian 进行测试： ➨ `https:// www.debian.org/devel/debian-installer/` 如果你能在这个虚拟机上确认这个问题，那么你就可以把这个描述直接提交给 Debian，并且把这个虚拟机附在这个报告后面。

大多数有关应用程序行为的 bug 报告应该直接报告给上游开发者，除非遇到集成问题。在这种情况下，错误是软件被打包并集成到 Debian 或 Kali 中的过程中发生的。例如，如果应用程序提供了编译选项，而程序包未启用，或者应用程序由于缺少库而无法工作（因此忽略了包信息导致依赖不完整），则可能面临集成问题。当你不知道你面对的是什么样的问题时，通常最好把问题提交给双方，并做交叉验证。

获取上游项目提交 bug 报告的网址通常并不复杂。只需要找到安装包元数据的 Homepage 字段中引用的上游网站即可：

```
$ dpkg -s sparta | grep ^Homepage:
Homepage: https://github.com/SECFORCE/sparta
```

6.3.3 如何填写一个bug报告

填写一个Kali的bug报告

Kali 在 `http://bugs.kali.org` 上搭建了一个基于 Web 的 bug 跟踪器，你可以匿名查阅所有 bug 报告，但是如果你想评论或提交新的 bug 报告，就需要去注册一个账户了。

注册一个 bug 跟踪器账户

首先，请在 bug 跟踪器网站上单击 signup 来注册一个新用户，如图 6.1 所示。

图6.1　Kali bug 跟踪器开始页面

接下来，提供用户名、电子邮件地址，并对验证码进行验证，然后单击 Sigup 按钮继续（如图 6.2 所示）。

如果注册成功，下一页（如图 6.3 所示）将通知你账户注册已经处理完毕，bug 跟踪系统将发送确认电子邮件到你提供的地址。你需要单击电子邮件中的链接才能激活你的账户。

激活账户后，单击 Proceed 链接进入 bug 跟踪器登录页面。

创建报告

在开始提交 bug 报告之前，请先登录你的账户，然后单击登录页上的报告问题链接。你将看到一个包含许多字段的表单，如图 6.4 所示。

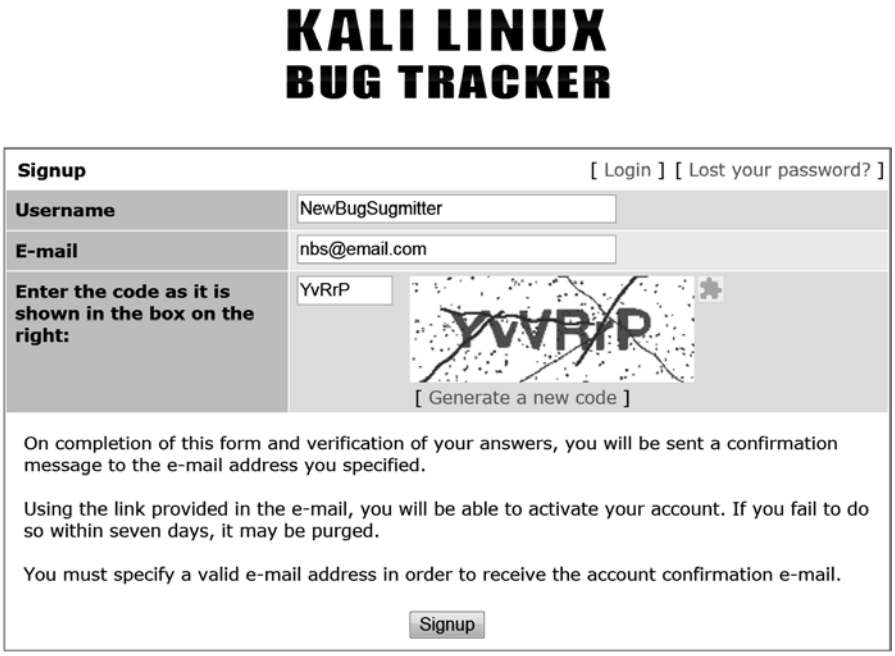


图6.2　注册页面

图6.3　注册确认页

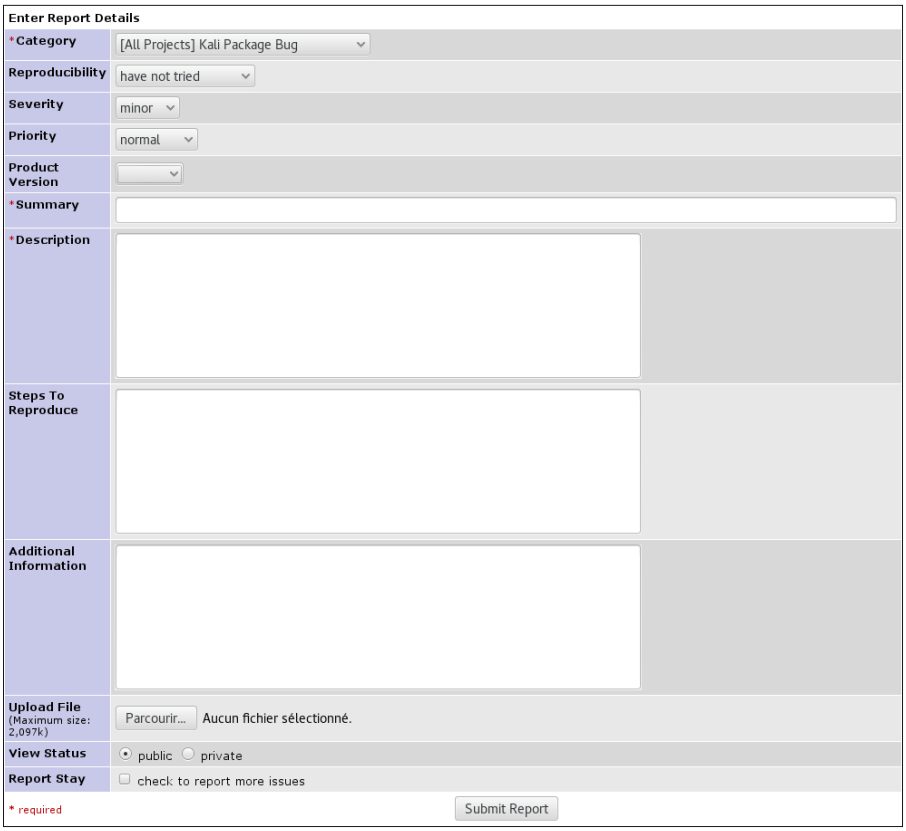


图6.4　报告bug表单

以下是表单上所有字段的简要说明。

Category（类别）（必填）

该字段描述你提交错误的类别。如果归因于特定安装包则可以选择 kali package bug 或 kali package improvement 类别。其他报告可以使用常规错误或功能请求问题的类别。其余的类别是针对特定用例的。tool upgrade 可以用来通知 Kali 开发者在 Kali 中打包新版本软件的可用性。New Tool Requests 可用于建议将新工具打包并集成到 Kali 发行版中。

Reproducibility（复现性）

这个字段记录问题是否以可预测的方式复现，或者是否只是随机发生。

Severity 和 Priority（严重性和优先级）

这些字段最好不填写，因为它们主要是为开发者准备的。他们可以根据问题的严重性和

处理的优先级，来排序问题清单。

Product Version（产品版本）

这个字段应该指出你正在运行的 Kali Linux 的版本（或者指出你需要更新的下一版本）。在报告关于旧版本的问题之前，会提示你再三考虑，因为对方可能已经不对旧版本提供支持了。

Summary（摘要）（必填）

这基本上可以说是错误报告的标题，这是人们首先会看到的。确保它传达了提交报告的原因。避免像“X 不工作”这样的泛泛性的描述，而应该选择“X 在条件 Z 下失败，出错 Y”这样的描述。

Description（说明）（必填）

这是报告的主体。在这里，你应该输入所收集的关于遇到问题的所有信息。不要忘记前一节给出的任何建议。

Steps to Reproduce（复现步骤）

在此字段中，填写有关如何触发问题的所有详细说明。

Additional Information（其他信息）

在这里，你可以提供你认为与问题相关的任何其他信息。如果你有问题的解决方法或变通方案，请在这里提供。

Upload File（上传文件）

并不是所有的东西都可以用文本来解释。在这里将任意文件附加到报告中，如显示错误的屏幕截图、触发问题的示例文档和日志文件等。

View Status（查看状态）

将该字段设置为 public，以便每个人都可以看到你的错误报告。仅将包含未公开安全漏洞信息的安全相关报告设置为 private。

在Debian中提交一个错误报告

Debian 使用一个基于电子邮件的 bug 跟踪系统，称为 Debbugs。要开启一个新的错误报告，你需要发送一封电子邮件给 `submi@bugs.debian.org`。它将给你分配一个错误号 XXXXXX，并通知你可以通过给 XXXXX@bugs.debian.org 发送邮件补充更多信息。每个 bug 都与 Debian 软件包相关联。你可以浏览 https://bugs.debian.org/package 中给出的关于软件包的所有错误（包括你正在跟进的错误报告）。你也可以在 https://bugs.debian.org/XXXXX 查看给定错误报告的历史记录。

设置 Reportbug

虽然你可以简单地使用电子邮件开启一个新的错误报告，但我们建议使用 reportbug 这个工具，因为它可以帮助你编写一个包含所有必需信息的完整错误报告。

理想情况下，你应该从 Debian 系统启动这个工具（例如，在你复现问题的虚拟机中）。

首次运行 reportbug 会启动一个配置脚本。首先，要我们选择一个技能等级。你应该选择新手或标准。我们使用后者，因为它提供了更细致的控制。而后，来到下一个界面并输入你的个人信息。最后，选择一个用户接口。配置脚本将帮你设置本地邮件传输代理、SMTP 服务器或 Debian SMTP 服务器。

```
Welcome to reportbug! Since it looks like this is the first time you have
used reportbug, we are configuring its behavior. These settings will be
saved to the file "/root/.reportbugrc", which you will be free to edit
further.
Please choose the default operating mode for reportbug.

1 novice    Offer simple prompts, bypassing technical questions.

2 standard  Offer more extensive prompts, including asking about things that
            a moderately sophisticated user would be expected to know about
            Debian.

3 advanced  Like standard, but assumes you know a bit more about Debian,
     ➥ including "incoming".

4 expert    Bypass most handholding measures and preliminary triage routines.
            This mode should not be used by people unfamiliar with Debian's
            policies and operating procedures.

Select mode: [novice] standard
Please choose the default interface for reportbug.

1 text   A text-oriented console user interface

2 gtk2   A graphical (GTK+) user interface.

3 urwid  A menu-based console user interface

Select interface: text
Will reportbug often have direct Internet access? (You should answer yes to
this question unless you know what you are doing and plan to check whether
duplicate reports have been filed via some other channel.)
```

```
[Y|n|q|?]? Y
What real name should be used for sending bug reports?
[root]> Raphaël Hertzog
Which of your email addresses should be used when sending bug reports?
(Note that this address will be visible in the bug tracking system, so you
may want to use a webmail address or another address with good spam
filtering capabilities.)
[root@localhost.localdomain]> buxy@kali.org
Do you have a "mail transport agent" (MTA) like Exim, Postfix or SSMTP
configured on this computer to send mail to the Internet? [y|N|q|?]? N
Please enter the name of your SMTP host. Usually it's called something like
"mail.example.org" or "smtp.example.org". If you need to use a different
port than default, use the : alternative format. Just press ENTER if you
don't have one or don't know, and so a Debian SMTP host will be used.
>
Please enter the name of your proxy server. It should only use this
parameter if you are behind a firewall. The PROXY argument should be
formatted as a valid HTTP URL, including (if necessary) a port number; for
example, http://192.168.1.1:3128/. Just press ENTER if you don't have one or
don't know.
>
Default preferences file written. To reconfigure, re-run reportbug with the
"--configure" option.
```

使用 reportbug

在完成设置后，可以开始进行实际的错误报告编写工作。系统会提示你输入软件包名称，你也可以直接在命令行上使用 `reportbug package`，来直接添加软件包名。

```
Running 'reportbug' as root is probably insecure! Continue [y|N|q|?]? y
Please enter the name of the package in which you have found a problem, or
type 'other'
to report a more general problem. If you don't know what package the bug is
in, please
contact debian-user@lists.debian.org for assistance.
> wireshark
```

与上面例子给出的建议相反，如果不知道该针对哪个软件包来提交错误，你应该联系 Kali 支持论坛（参见 6.2 节）。下一步，reportbug 会下载与包相对应的错误列表（其他人已经提交的错误列表），并让你浏览它们，以查看是否可以找到与你一样的错误。

```
*** Welcome to reportbug.  Use ? for help at prompts. ***
Note: bug reports are publicly archived (including the email address of the
```

```
submitter).
Detected character set: UTF-8
Please change your locale if this is incorrect.

Using '"Raphaël Hertzog" <buxy@kali.org>' as your from address.
Getting status for wireshark...
Verifying package integrity...
Checking for newer versions at madison...
Will send report to Debian (per lsb_release).
Querying Debian BTS for reports on wireshark (source)...
35 bug reports found:

Bugs with severity important
   1) #478200  tshark: seems to ignore read filters when writing to…
   2) #776206  mergecap: Fails to create output file > 2GB
   3) #780089  wireshark: "On gnome wireshark has not title bar. Does…
Bugs with severity normal
   4) #151017  ethereal: "Protocol Hierarchy Statistics" give misleading…
   5) #275839  doesn't correctly dissect ESMTP pipelining
[...]
  35) #815122  wireshark: add OID 1.3.6.1.4.1.11129.2.4.2
(24-35/35) Is the bug you found listed above [y|N|b|m|r|q|s|f|e|?]? ?
y - Problem already reported; optionally add extra information.
N - (default) Problem not listed above; possibly check more.
b - Open the complete bugs list in a web browser.
m - Get more information about a bug (you can also enter a number
    without selecting "m" first).
r - Redisplay the last bugs shown.
q - I'm bored; quit please.
s - Skip remaining problems; file a new report immediately.
f - Filter bug list using a pattern.
e - Open the report using an e-mail client.
? - Display this help.
(24-35/35) Is the bug you found listed above [y|N|b|m|r|q|s|f|e|?]? n
Maintainer for wireshark is 'Balint Reczey <balint@balintreczey.hu>'.
Looking up dependencies of wireshark...
```

如果你发现你的 bug 已经在列表中存档了，可以选择发送补充信息，如不在列表中，你将被要求提交新的错误报告：

```
Briefly describe the problem (max. 100 characters allowed). This will be
the bug email subject, so keep the summary as concise as possible, for
example: "fails to send email" or "does not start with -q option
```

```
specified" (enter Ctrl+c to exit reportbug without reporting a bug).
> does not dissect protocol foobar
Rewriting subject to 'wireshark: does not dissect protocol foobar'
```

在用一句话描述了你的问题之后，你必须对其严重程度进行扩展描述：

```
How would you rate the severity of this problem or report?

1 critical       makes unrelated software on the system (or the whole system)
                 break, or causes serious data loss, or introduces a
                 security hole on systems where you install the package.
2 grave          makes the package in question unusable by most or all users,
                 or causes data loss, or introduces a security hole allowing
                 access to the accounts of users who use the package.
3 serious        is a severe violation of Debian policy (that is, the
                 problem is a violation of a 'must' or 'required' directive);
                 may or may not affect the usability of the package. Note
                 that non-severe policy violations may be 'normal,' 'minor,'
                 or 'wishlist' bugs. (Package maintainers may also designate
                 other bugs as 'serious' and thus release-critical; however,
                 end users should not do so.). For the canonical list of
                 issues worthing a serious severity you can refer to this
                 webpage: http://release.debian.org/testing/rc_policy.txt
4 important      a bug which has a major effect on the usability of a
                 package, without rendering it completely unusable to
                 everyone.
5 does-not-build  a bug that stops the package from being built from source.
                  (This is a 'virtual severity'.)
6 normal         a bug that does not undermine the usability of the whole
                 package; for example, a problem with a particular option or
                 menu item.
7 minor          things like spelling mistakes and other minor cosmetic
                 errors that do not affect the core functionality of the
                 package.
8 wishlist       suggestions and requests for new features.

Please select a severity level: [normal]
```

如果你不确定，请保持 Normal 默认的严重性级别。

你还可以使用几个关键字标记你的报告：

```
Do any of the following apply to this report?
```

```
1 d-i        This bug is relevant to the development of debian-installer.
2 ipv6       This bug affects support for Internet Protocol version 6.
3 l10n       This bug reports a localization/internationalization issue.
4 lfs        This bug affects support for large files (over 2 gigabytes).
5 newcomer   This bug has a known solution but the maintainer requests someone
             else implement it.
6 patch      You are including a patch to fix this problem.
7 upstream   This bug applies to the upstream part of the package.
8 none

Please select tags: (one at a time) [none]
```

大多数标签是相当难懂的，但是如果你的报告包含修复内容，你应该选择 PATCH 标签。

完成此操作后，reportbug 将打开一个文本编辑器，其中包含可以参考的编辑模板（见示例 6.2）。它包含一些你应该删除和回答的问题，以及一些已经自动收集的有关系统的信息。注意前几行的结构，通常不应该修改它们，因为它们将直接被 bug 跟踪器解析，以便将报告分配给正确的软件包。

示例 6.2　由 reportbug 工具生成的模板

```
Subject: wireshark: does not dissect protocol foobar

Package: wireshark
Version: 2.0.2+ga16e22e-1
Severity: normal

Dear Maintainer,

*** Reporter, please consider answering these questions, where appropriate
***

   * What led up to the situation?
   * What exactly did you do (or not do) that was effective (or
     ineffective)?
   * What was the outcome of this action?
   * What outcome did you expect instead?

*** End of the template - remove these template lines ***


-- System Information:
Debian Release: stretch/sid
  APT prefers testing
```

```
  APT policy: (500, 'testing')
Architecture: amd64 (x86_64)
Foreign Architectures: i386

Kernel: Linux 4.4.0-1-amd64 (SMP w/4 CPU cores)
Locale: LANG=fr_FR.utf8, LC_CTYPE=fr_FR.utf8 (charmap=UTF-8)
Shell: /bin/sh linked to /bin/dash
Init: systemd (via /run/systemd/system)

Versions of packages wireshark depends on:
ii  wireshark-qt  2.0.2+ga16e22e-1

wireshark recommends no packages.

wireshark suggests no packages.

-- no debconf information
```

保存报告并关闭文本编辑器后，返回 reportbug，其将提供许多其他选项，并帮你把生成好的报告发出去。

```
Spawning sensible-editor...
Report will be sent to "Debian Bug Tracking System" <submit@bugs.debian.org>
Submit this report on wireshark (e to edit) [Y|n|a|c|e|i|l|m|p|q|d|t|s|?]? ?
Y - (default) Submit the bug report via email.
n - Don't submit the bug report; instead, save it in a temporary file (exits
    reportbug).
a - Attach a file.
c - Change editor and re-edit.
e - Re-edit the bug report.
i - Include a text file.
l - Pipe the message through the pager.
m - Choose a mailer to edit the report.
p - print message to stdout.
q - Save it in a temporary file and quit.
d - Detach an attachment file.
t - Add tags.
s - Add a X-Debbugs-CC recipient (a CC but after BTS processing).
? - Display this help.
Submit this report on wireshark (e to edit) [Y|n|a|c|e|i|l|m|p|q|d|t|s|?]? Y
Saving a backup of the report at /tmp/reportbug-wireshark-backup-20160328-
19073-87oJWJ
Connecting to reportbug.debian.org via SMTP...
```

```
Bug report submitted to: "Debian Bug Tracking System"
<submit@bugs.debian.org>
Copies will be sent after processing to:
  buxy@kali.org

If you want to provide additional information, please wait to receive the
bug tracking number via email; you may then send any extra information to
n@bugs.debian.org (e.g. 999999@bugs.debian.org), where n is the bug
number. Normally you will receive an acknowledgement via email including
the bug report number within an hour; if you haven't received a
confirmation, then the bug reporting process failed at some point
(reportbug or MTA failure, BTS maintenance, etc.).
```

在另一个自由软件项目中填写bug报告

各式各样的自由软件，它们都使用不同的工作流和处理工具。对于这些多样性的自由软件也需要使用 bug 跟踪器。尽管大多数的自由软件项目都在 GitHub 上托管，并可以使用 GitHub 来跟踪处理它们的 bug，但也有许多其他的托管站点，它们使用基于 Bugzilla、Trac、Redmine、Flyspray 等的 bug 跟踪器。这些跟踪器大多数都是基于网络的，并要求你注册一个账户以提交一份新的报告。

我们不会在这里覆盖所有的 bug 跟踪器。你可以自己学习其他开源软件项目的各种跟踪器，但是由于 GitHub 比较流行，我们在这里对其作以简单介绍。与其他跟踪器一样，你必须先创建一个账户并登录。接下来，单击 Issues 选项卡，如图 6.5 所示。

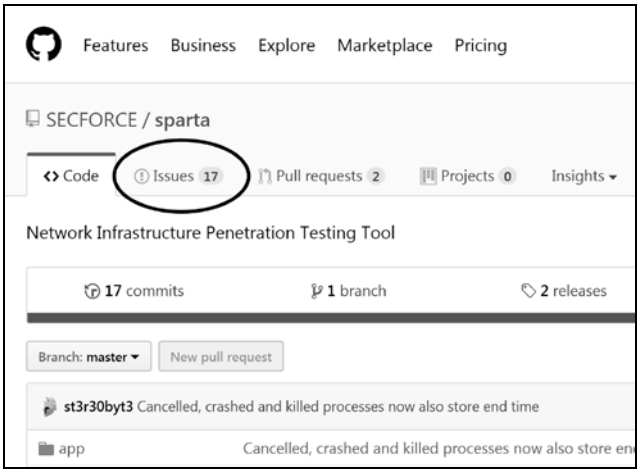


图6.5　GitHub项目的主页面

在这里可以浏览（并搜索）未解决问题的列表。一旦确信你的 bug 没有被他人提交，你可以单击 New issue 按钮（如图 6.6 所示）。

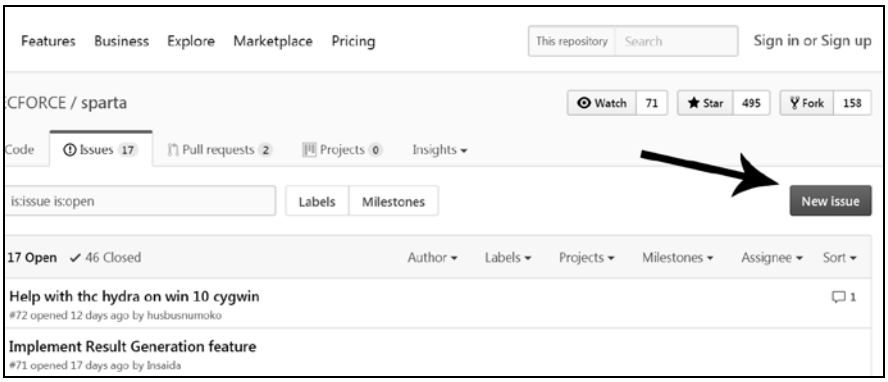


图6.6　GitHub项目的问题页面

你可以在该页面上描述你的问题（如图 6.7 所示）。虽然不像 reportbug 中的模板，但是此处的漏洞报告机制相当直接，你可以附加文件，编写文本等。当然，为了获得最佳效果，务必按照我们之前讨论的书写原则，以创建详尽有效的错误报告。

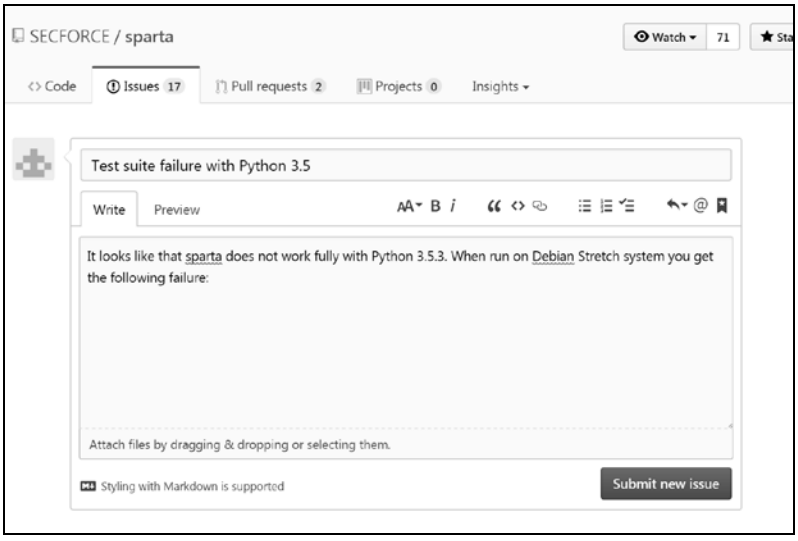


图6.7　提交新问题的GitHub表单

6.4　小结

在本章中，我们讨论了各种方法，帮助你查找有关程序的文档和信息，以及如何找到可能遇到问题的帮助信息。我们了解了 man 和 info 页面，以及 `apropos` 和 `info` 命令。讨论了 bug 跟踪器，给出了一些关于如何搜索和提交 bug 报告的提示，并介绍了一些技巧来帮助你定位相关程序或项目。

要点提示:

- 在发现问题之前，需要了解每个程序的作用，然后才能理解发生问题时的真实情况。最好的方法就是审查程序的每个文件。
- 查看手册页，只需键入 `man manual-page`，然后在可选的章节号后面填入命令名称。
- `apropos` 命令返回手册页中命令描述内容符合查询关键字的数据列表，以及手册页中有关的单行摘要。
- GNU 项目为大部分程序都编写了 info 格式的手册页。这就是为什么许多手册页都会去引用相应的 info 文档的原因。
- 每个软件包内都含有文档，哪怕最简单的文档程序通常都会包含 README 文件。这些文档通常安装在/usr/share/doc/package/目录中。
- 在大多数情况下，查看程序官方网站的 FAQ 或邮件联系相关方面可能会解决你遇到的问题。
- Kali Linux 项目在 `http://docs.kali.org` 上部署了一系列非常有用的文档。
- Kali Linux项目使用Freenode[1] IRC网络平台上的#kali-linux频道。你可以使用chat.freenode.net作为IRC服务器，在端口 6667 上进行TLS加密的连接，也可以在端口 6666 上进行明文连接。要加入IRC平台上的频道讨论，你必须使用诸如hexchat（以图形模式）或irssi（以控制台模式）的IRC客户端，webchat.freenode.net[2]上还有一个基于Web的客户端。
- Kali Linux项目官方社区论坛位于forums.kali.org[3]。
- 如果你发现程序中有错误，你可以搜索错误报告或自行归档上报该问题。务必遵循我们概述的指导原则，以确保你的报告清晰、全面，从而提高问题被开发人员及时处理

1 https://www.freenode.net
2 https://webchat.freenode.net
3 https://forums.kali.org

的可能性。

- 有的错误报告应提交给 Kali，有一些则可能给 Debian 方提交更加合理。输入命令 `dpkg -s package-name | grep ^ Version`，显示软件的版本号，如果它是 Kali 修改的包，将被标记为“kali”。
- 识别上游项目并找到提交错误报告方通常很简单。只需查看安装包元数据 Homepage 字段中引用的上游网站地址就可以了。
- Kali 在 `https://bugs.kali.org` 上使用基于 Web 的错误跟踪器，你可以匿名查看所有错误报告，但是如果你想评论或提交新的错误报告，则需要注册一个账户。
- Debian 使用的是（大部分）基于电子邮件的 bug 跟踪系统，称为 Debbugs。要开启一个新的错误报告，你需要发送一封电子邮件（使用特殊语法）到 submit@bugs.debian.org，或者你可以使用 `reportbug` 命令，它将指导你完成整个过程。
- 尽管许多项目都在 GitHub 上托管，并使用 GitHub 上的问题来跟踪 bug，但也有许多使用其他托管跟踪器的项目。如果需要给第三方错误跟踪器发布，你可能需要研究其基础用法。

现在，我们已经拥有了用于 Linux 操作，安装和配置 Kali，以及解决系统问题获得帮助的基础知识和工具，下一章我们将讨论如何来保护你安装的服务和系统上的各种数据。

练习题

练习1——Kali源

1．如果你想知道$xyz 之后更新版本的 nmap 软件是否已在 Kali 中包含，通过什么资源是最快捷的？

2．两个最主要的 Kali 官方资源社区是什么？

3．如何搜索与特定关键字相关的手册页？

第7章　加固和监控Kali Linux

内容

- 定义安全策略
- 可用的安全措施
- 对网络服务进行安全加固
- 防火墙和包过滤
- 监控和日志
- 小结

当使用 Kali Linux 处理较为敏感和重要的工作时，你可能需要更加重视安全方面的设置。在本章中，我们首先讨论安全策略，介绍在定义此类策略时需要考虑的各个要点，并概述可能威胁你个人隐私或者系统安全的风险。我们还将讨论关于笔记本电脑与个人桌面电脑的相关安全措施，并重点介绍防火墙和包过滤。最后，讨论监控工具和相关策略，并描述如何使用它们检测系统的潜在威胁。

7.1　定义安全策略

大范围地讨论安全是不切实际的想法，因为没有哪个概念、工具和程序，能够普遍适用。你需要确切地知道自己的需求是哪些，并从中做出选择。让我们从回答几个问题开始。简单随意地配置工具，可能会引起很多安全方面的问题。

通常，我们最好确定一个具体的目标。一个好的实施方案通常需要考虑以下问题：

- 我们要保护的目标是什么？安全策略会由于你想保护的目标不同（例如保护系统还是数据）而有所不同。如果你想保护的是数据，还需要知道要保护哪些数据。
- 我们要防止的事情是什么？是数据泄露么？还是数据丢失？还是拒绝服务导致经济损失？
- 还有就是你要对抗的是谁？对内部常规用户和对外部的攻击者是完全不同的策略。

“风险”这个词通常指向这三个因素：保护什么？应该避免什么？谁可能会让这种事发生？想要建立一个风险控制模型，就一定要弄清楚这三个问题。通过建立风险控制模型，来构建一个安全策略，并且通过设计具体方案来实施安全策略。

永恒的话题

布鲁斯·施奈尔，一位全球著名的网络安全专家（不仅在计算机领域）提过一个关于安全的非常重要的理念：安全是一个过程，而不是一个产品。需要被保护的资产随着时间推移一直在变化，风险也是一样，那么对于潜在的攻击者来说，攻击手段也是在不断变化的。即便你已经有一个设计完美的安全策略并已实施，你也不能有一刻的放松。因为风险在不断发展，你应对风险的策略也一定要进行相应的调整。

也需要将额外的约束考虑在内，因为它们会限制可用策略的适用范围。你希望这个系统获得什么级别的安全保护？这个问题又将会对如何实现安全策略产生很大影响。虽然最终方案常常是根据整个安全策略的预算来制定的，但其他元素也应该考虑在内，比如由于安全策略给用户带来的不便，或者带来的系统性能消耗等。

风险对象确定好之后，我们就可以着手考虑如何设计可行的安全策略了。

在你决定采用哪种安全保护策略时，需要考虑到一些极端情况。它可以是非常简单的，只提供最基本的系统安全设置。

例如，对于一台只在每天下班时往其中添加一些不太重要数据的旧电脑，在考虑它的安全策略时，也许什么都不做才是最合理的设计。对于本身价值很低，又没有存储任何数据的计算机，即便被黑客控制了也就是多了一个节点而已，但是你维护并对其进行安全架构设计与防护的代价，可能远远超出了攻击它所能产生的价值。

从另一个极端来说，考虑到各种可能的问题。你可能希望机密数据能够得到最好的保护，在这种情况下，一定要有一个完备的处理销毁文件的设计（安全地删除文件，擦除硬盘的各个 bit 位，用浓酸物理溶解销毁磁盘等）。如果还有将这些数据保存起来以供将来使用（虽然不一定可以随时取用）的需求，如果不计成本的话，那么可以将数据存储在用铱铂合金制作的磁盘上，放置在各种各样山脉之下的防弹掩体内，并且在完全秘密的情况下用一支军队把守各个出入口。

尽管这些例子看起来过于极端了，它们不过是一些对应某些风险定义的解决方案，需要根据实际安全需求和客观存在的各种限制，来进行综合性的考虑和设计。最终设计策略应该是最合理的，而不是最好，或者最差的，或者其他的什么。

回到一个更典型的案例，一个信息系统可以被划分为通用系统和独立子系统。每个子系统都有自己的需求和约束条件。因此，风险评估和安全策略都需要单独设计。有一个要牢记的原则是，保护一个小的暴露面要比维护大的容易得多。网络结构也应该遵循这个原则设

计：敏感的服务应该集中在少量的机器上，这些机器只能通过最小数量的通道进行登录或检查。它的逻辑也很简单：只保护几个点要比保护所有暴露在互联网上的设备容易得多。基于这一点，网络过滤设备（包括防火墙）的价值变得明显了。这些防护设备可以使用专用硬件来实现，但一个更简单和更灵活的解决方案，就是使用一个集成好的软件防火墙，如 Linux 系统。

7.2　可用的安全措施

就像前面介绍的，关于如何安全使用和设置 Kali Linux 没有一个固定的方法，关键看你如何使用和你想保护什么。

7.2.1　服务器

如果你选择 Kali Linux 作为一个服务器的操作系统，你最先想到的应该是修改那些存在默认口令的服务，来进行加固（参见 7.3 节的内容），并且调整防火墙的访问控制策略（参见 7.4 节的内容）。

如果你直接在服务器或者某个服务设立了一个账户，你一定要确保为该账户设置了强密码（可以抵抗暴力破解攻击）。与此同时，你可以通过设置 fail2ban，增大攻击者通过网络破解密码的难度（通过限制多次尝试登录失败的 IP 地址）。可以通过命令 `apt update`，然后运行 `apt install fail2ban`，来完成 fail2ban 的安装。

如果你运行的是 Web 服务，你可能还想利用 HTTPS 加密网站通信过程，以防止被网络嗅探（流量里可能存在用户 cookie）。

7.2.2　个人电脑

渗透测试人员的笔记本电脑肯定不会跟公共服务器面临同样的风险。例如，笔记本电脑不太可能受到来自脚本小子的随机扫描攻击，而且即使你正在使用，电脑上也不太会开放什么网络服务。

真正的风险通常是在电脑从一个状态转移到另一个状态的时候产生的。例如，你的笔记本电脑在旅行时可能被偷或者被海关扣下。这就是为什么你要考虑对电脑进行全盘加密的原因（见 4.2.2 节的内容），并给整个磁盘设置超强口令来对其加密，特别是那些你还在收集和

处理的机密数据，更需要得到最大限度的保护。

你可能还需要调整防火墙规则（参见 7.4 节的内容），另外对于个人电脑的安全需求肯定和服务器是不一样的。你可能想要禁止除指定 VPN 链路外的所有其他对外流量。这个策略可以保证当前计算机处于一个安全网络中，一旦 VPN 链路出现问题，你马上就能注意到（而不会有流量通过本地网络流出）。这样你就不会在浏览网页或者在其他线上操作时暴露你的 IP 地址了。此外，如果你正在执行一个内网的渗透测试任务，最好保证所有网络活动都处在你的控制范围内，并减少网络噪音，以免引起其他网路用户的注意，或者触发防御系统。

7.3 对网络服务进行安全加固

通常情况下，把你不用的服务都关掉是一个好习惯。Kali 已经帮助大家做到了这一点，默认已经关闭了所有的网络服务。

只要这些服务处于禁用的状态，它们就不会引发什么安全隐患。因此，每当你要启用某个服务的时候，一定要加倍小心，因为：

在默认情况下是没有开启防火墙的，所以如果有人监听了所有的网络接口，这些服务一旦开启就可以被他们访问到。

有些服务默认是没有做身份验证的，只有在第一次使用时才会设置；有的服务是有默认密码的（被大家熟知）。所以启用前，确保你已经把默认密码设置成只有你才知道的强密码。

在这里我们要强调一下，很多服务是以 root 角色运行的并且拥有完整的管理员权限，所以未授权访问和安全漏洞是非常危险的，可能会造成严重影响。

默认身份验证

这里不会列出所有默认可做身份验证的服务，而是推荐大家自己去查看readme.debian文件，或者登录`docs.kali.org`[1]和`tools.kali.org`[2]，看看都有哪些服务需要做进一步步安全加固。

如果你以自生模式运行 Kali，那么 root 密码就应该是“toor”。因此在修改默认密码或者禁用远程登录前，不要开启 SSH 服务。

还要提醒一点，BeEF 程序（默认已经安装了 beef-xss 安装包）也有一个被大众所知的用户密码组合，user：“beef”；password：“beef”，并已经被写死在它的配置文件里。

1 https://docs.kali.org
2 https://tools.kali.org

7.4　防火墙和包过滤

防火墙可以是电脑硬件、软件或者两者相结合的一种计算机设备，它们分析输入或输出网络数据包（发向或离开本地网络），并只允许符合限制条件的流量包通过。

过滤网关就是一种保护整个网络的防火墙。它通常被安装配置成为网络的网关，以便它可以解析所有进出网络的数据包。另外还有一种本地防火墙，它们通过在主机本地部署一个软件服务，来控制该主机和主机上服务的网络进出流量，或者来限制一些未知恶意软件主动向外部网络的连接或发送数据。

Linux 的内核中已经嵌入了 netfilter 防火墙。由于网络环境和用户的需求不同，所以并没有一个已经集成好的通用防火墙设置解决方案。但是，你可以使用 `iptables` 和 `ip6tables` 命令来控制 netfilter。这两个命令之间的区别是前者适用于 IPv4 网络，而后者适用于 IPv6。由于两种网络协议栈已经被使用多年了，所以这两种工具需要根据实际情况结合使用。你还可以使用一个名为 `fwbuilder` 的很不错的工具，它提供了可视化的过滤规则视图。

7.4.1　netfilter 行为

netfilter 通过 4 张不同的表来存储规则，并控制 3 种针对数据包的不同操作：

- `filter` 用于过滤规则处理（接受、拒绝或忽略数据包）。
- `nat`（Network Address Translation）关注数据包的地址和端口的转换。
- `mangle` 设置或改变数据包的服务类型（包括 ToS-Type of Service 字段和选项）。
- `raw` 允许在数据包到达网络连接跟踪系统前进行其他手动修改。

每个表都包含称为链的规则列表。防火墙根据预定义的情况，使用标准链来处理数据包。管理员可以创建其他链，该链只有在被标准链引用（直接或间接）时才会生效。过滤规则表有三个标准链。

- `INPUT`：针对目的地的防火墙本身的数据包。
- `OUTPUT`：处理防火墙自己产生发送出去的数据包。
- `FORWARD`：关注通过防火墙的数据包（防火墙既不是其源也不是目的地）。

nat 规则表也有三个标准链：

- `PREOUTING`：接收到包后将其地址修改成内网地址。
- `POSTROUTING`：要发送包时将其地址修改为对外地址。

- OUTPUT：修改防火墙自身产生的数据包。

图 7.1 显示了这些链表的调用流程。

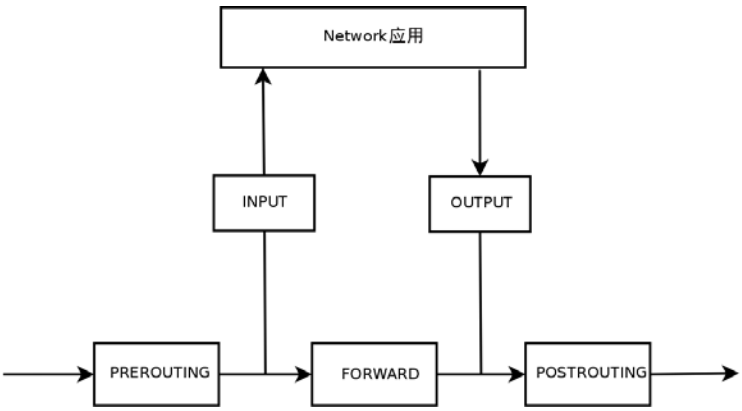


图7.1　netfilter 链表的调用流程

每个链都是一条规则表，每个规则由一组条件和满足条件时执行的操作组成。当处理数据包时，防火墙逐一扫描并尝试匹配链，当满足一个规则的条件时，它会跳转（因此在命令中使用-j 选项）到指定操作继续处理。大多数常用行为都已经被标准化了，并为它们设计了专门的操作。使用标准操作会中断对链的处理过程，因为符合标准操作的数据包处理过程已经被封装好了。Netfilter 的操作如下所示。

- ACCEPT：允许数据包通过。
- REJECT：拒绝具有 ICMP 错误报文的报文（命令 iptables 的--reject-with type 选项可以设定错误类型）。
- DROP：删除（忽略）数据包。
- LOG：记录（通过 syslogd 命令）具有数据包描述的消息。注意这个行为不会中断处理，并且在下一个规则匹配时继续执行链，这就是为什么记录拒绝数据包时既需要记录规则，也需要拒绝 / 丢弃规则的原因；与记录相关的参数包括：
  - log-level，默认值为 warning，表示 syslog 严重性级别。
  - log-prefix，允许指定文本前缀来区分记录的消息。
  - log-tcp-sequence，--log-tcp-options 和 log-ip-options 将相关数据额外的信息整合到日志中，它们分别对应 TCP 序列号、TCP 选项和 IP 选项。
- ULOG：通过 ulogd 记录信息，在处理大量信息时，它比 syslogd 更合适有效。注意，该规则类似 LOG，处理下一条规则时也会返回到调用链的下一条规则。

- CHAIN_NAME：跳转到指定链并处理其规则。
- RETURN：中断当前链的处理过程，并返回调用链；如果当前链是一个标准链，并且没有调用链，默认（由 iptables 中-P 选项定义）就会执行该链。
- SNAT（仅在 nat 表中）：源网络地址转换（SNAT）。通过附加选项描述了应用的确切变化，包括--to-source 选项，定义新的源 IP 地址和/或端口。
- DNAT（仅在 nat 表中）：目标网络地址转换（DNAT）。通过附加选项描述了应用的确切变化，包括--to-destination address:port 选项，定义新的目的地 IP 地址和端口。
- MASQUERADE（仅在 nat 表中）：地址伪装（SourceNAT 的一种特殊应用）。
- REDIRECT（仅在 nat 表中）：将数据包重定向到防火墙指定的端口。可把端口透明映射到一个代理服务器的服务端口上，而客户端不用做任何配置上的设置，客户端在连接时就好像直接连接到代理服务端一样。--to-ports ports 选项表示端口或端口范围，用来对其中的数据包进行重定向。

还有一些其他功能，特别对于 mangle 表，超出了本书讨论的范围。在 iptables（8）和 ip6tables（8）的手册页有全面的介绍。

什么是 ICMP

互联网控制消息协议（ICMP）是一种用于传输通信辅助信息的协议。它使用 ping 命令测试网络连接，ping 命令发送一个 ICMP 回显请求消息，接收方用 ICMP 回显应答消息来回答。它可以用于告诉防火墙拒绝某些数据包，帮助缓解接收缓冲区溢出问题，为网络连接中的下一个数据包提供更好的路由服务，等等。这个协议是由几个 RFC 文件定义的。RFC 777 和 RFC 792 便是最早的一批标准，但是许多其他标准对其进行了扩展和修改。

➨ http://www.faqs.org/rfcs/rfc777.html

➨ http://www.faqs.org/rfcs/rfc792.html

作为参考，接收缓冲区是一个小内存区域，用来存储已经从网络到达本地而内核还未来得及处理的数据。如果该区域满了，就不能接收新的数据，ICMP 会发出错误信号，这样发送者会减缓传输速率（理想情况下，过段时间就会达到平衡）。

注意，虽然 IPv4 在没有 ICMP 的情况下，仍然能工作，但是，ICMPv6 对于 IPv6 来说是必须的。因为它结合了很多在 IPv4 网络中被广泛使用的 ICMPv4 功能：IGMP（Internet Group Membership Protocol）和 ARP（Address Resolution Protocol）。ICMPv6 被定义在 RFC 4443 中。

➨ http://www.faqs.org/rfcs/rfc4443.html

7.4.2 iptables和ip6tables语法

`iptables` 和 `ip6tables` 命令可用于操作表、链和规则。其`-t table` 选项指定操作的是哪个表（默认为`filter`）。

命令

主要的链操作如下所示。

- `-L` chain 显示链中的规则。通常配合`-n` 选项，禁用机器名解析（例如，`iptables -n -L INPUT` 将显示与传入数据包相关的规则）。
- `-N` chain 创建新的用户自定义规则链。可以根据你的不同需求创建新链，包括测试网络服务状态或抵御网络攻击。
- `-X` chain 删除一个空的或者未使用的链（例如，`iptables -X ddos-attack`）。
- `-A` chain 在指定链的末尾添加一条规则。注意，规则的处理过程为从上到下，一定要在添加规则时考虑到这一点。
- `-I` chain rule_num rule 在规则号之前插入规则。和`-A` 选项一样，将新规则插入链中时，请牢记处理顺序。
- `-D` chain rule_num 或`-D` chin rule 删除链中的一条规则。在前条命令语法结构中，在参数后添加要被删除规则的号码（`iptables -L-line-numbers` 将显示这些规则的号码），而后者则通过规则内容来识别它。
- `-F` chain 清理一个链（删除其内所有的规则）。例如，删除所有处理传出数据包的规则，可以使用命令 `iptables -F output` 来实现。如果没有指定链，表中所有规则都将被删除。
- `-P` chain action 定义默认操作，或为指定链定义“策略”。注意，只有标准链可以被这样设置策略。例如要默认丢弃所有传入的流量，可以运行 `iptables -P input drop` 命令。

规则

每条规则都表示为 conditions -j action action_options 的形式。如果在同一条规则里描述了几个条件，那么各个条件之间的逻辑关系是“与”（and）关系，这至少比每个条件的限制要严格。

`-p` protocol 条件匹配 IP 数据包的协议区段。最常见的是 `tcp`、`udp`、`icmp` 和`icmpv6`。这个条件允许添加 TCP 端口来作为附加条件，如`--source-port port` 和`--`

destination-port port。

取反条件　在一个条件前添加感叹号会对这个条件取反。例如，在-p 选项之前加上感叹号表示求反，就变成了“除了指定协议之外的任何数据包”。这种取反机制不仅适用于 -p 选项，也可用于其他所有条件。

-s address 或者-s network/mask 条件匹配数据包的源地址。相对应的 -d address 或者 -d network/mask 用来匹配数据包的目的地址。

-i interface 条件选择来自指定网络接口的流入数据包。-o interface 选择特定接口上发出的数据包。

--state state 条件作用于连接中数据包状态（这个功能需要调用 ipt_conntrack 内核模块，用于网络连接跟踪）。新（new）状态说明这个包是用来启动一个新的会话连接的，ESTABLISHED 匹配属于现有连接的数据包，RELATED 匹配与现有连接相关的新连接数据包（常用于 FTP 协议“活动”模式下的连接）。

iptables 和 ip6tables 有许多可用的选项，掌握它们需要大量的学习和经验积累。最常使用的选项应该就是阻止某主机或主机群的恶意网络访问。例如，我们可以设置对来自 IP 地址 10.10.1.5 和 31.13.74.0/24 这个 C 段网络的流量都不响应：

```
# iptables -A INPUT -s 10.0.1.5 -j DROP
# iptables -A INPUT -s 31.13.74.0/24 -j DROP
# iptables -n -L INPUT
Chain INPUT (policy ACCEPT)
target    prot opt source            destination
DROP      all  --  10.0.1.5          0.0.0.0/0
DROP      all  --  31.13.74.0/24     0.0.0.0/0
```

另一个常用的 iptables 命令允许特定服务可以被网络访问。例如要允许用户访问主机上的 SSH、HTTP 和 IMAP 服务，可以运行以下命令：

```
# iptables -A INPUT -m state --state NEW -p tcp --dport 22 -j ACCEPT
# iptables -A INPUT -m state --state NEW -p tcp --dport 80 -j ACCEPT
# iptables -A INPUT -m state --state NEW -p tcp --dport 143 -j ACCEPT
# iptables -n -L INPUT
Chain INPUT (policy ACCEPT)
target     prot opt source            destination
DROP       all  --  10.0.1.5          0.0.0.0/0
DROP       all  --  31.13.74.0/24     0.0.0.0/0
ACCEPT     tcp  --  0.0.0.0/0         0.0.0.0/0      state NEW tcp dpt:22
ACCEPT     tcp  --  0.0.0.0/0         0.0.0.0/0      state NEW tcp dpt:80
```

```
ACCEPT     tcp  --  0.0.0.0/0          0.0.0.0/0       state NEW tcp dpt:143
```

清理那些已经弃用的和不必要的规则是良好的计算机使用习惯。删除 iptables 规则最简单的方法，是使用--line-numbers 命令获取要删除规则的行号，然后引用它来调整规则，不过我们要注意，删除一条规则后将重新编号链中后续的其他规则。

```
# iptables -n -L INPUT --line-numbers
Chain INPUT (policy ACCEPT)
num  target     prot opt source               destination
1    DROP       all  --  10.0.1.5            0.0.0.0/0
2    DROP       all  --  31.13.74.0/24       0.0.0.0/0
3    ACCEPT     tcp  --  0.0.0.0/0           0.0.0.0/0   state NEW tcp dpt:22
4    ACCEPT     tcp  --  0.0.0.0/0           0.0.0.0/0   state NEW tcp dpt:80
5    ACCEPT     tcp  --  0.0.0.0/0           0.0.0.0/0   state NEW tcp dpt:143
# iptables -D INPUT 2
# iptables -D INPUT 1
# iptables -n -L INPUT --line-numbers
Chain INPUT (policy ACCEPT)
num  target     prot opt source               destination
1    ACCEPT     tcp  --  0.0.0.0/0           0.0.0.0/0   state NEW tcp dpt:22
2    ACCEPT     tcp  --  0.0.0.0/0           0.0.0.0/0   state NEW tcp dpt:80
3    ACCEPT     tcp  --  0.0.0.0/0           0.0.0.0/0   state NEW tcp dpt:143
```

根据不同的情况，还可以设置很多过滤条件。更多信息，请参考 iptables（8）和 ip6tables（8）的说明文档。

7.4.3 创建规则

每创建一个规则都需要调用 iptables 或 ip6tables 命令。手动输入这些命令是非常烦琐的，所以我们通常调用一个事先编写好的脚本，使得机器每次启动后就可以进行同样的自动化配置。可以手写这个脚本，也可以使用诸如 fwbuilder 这样的高级工具来编写。

```
# apt install fwbuilder
```

编写方法很简单。第一步，列出所有要调用的规则元素：

- 防火墙自身以及网络接口
- 网络及其相应的 IP 范围
- 服务器
- 宿主服务器上托管服务的端口

然后可以选择对象并拖拽（drag-and-drop）来创建规则，如图 7.2 所示。可以在一些上下文菜单中改变条件（例如，取反）。然后选择操作并进行配置。至于 IPv6，可以为 IPv4 和 IPv6 分别创建规则集，或者只创建一个，然后使用 `fwbuilder` 来根据赋予对象的地址转换规则。

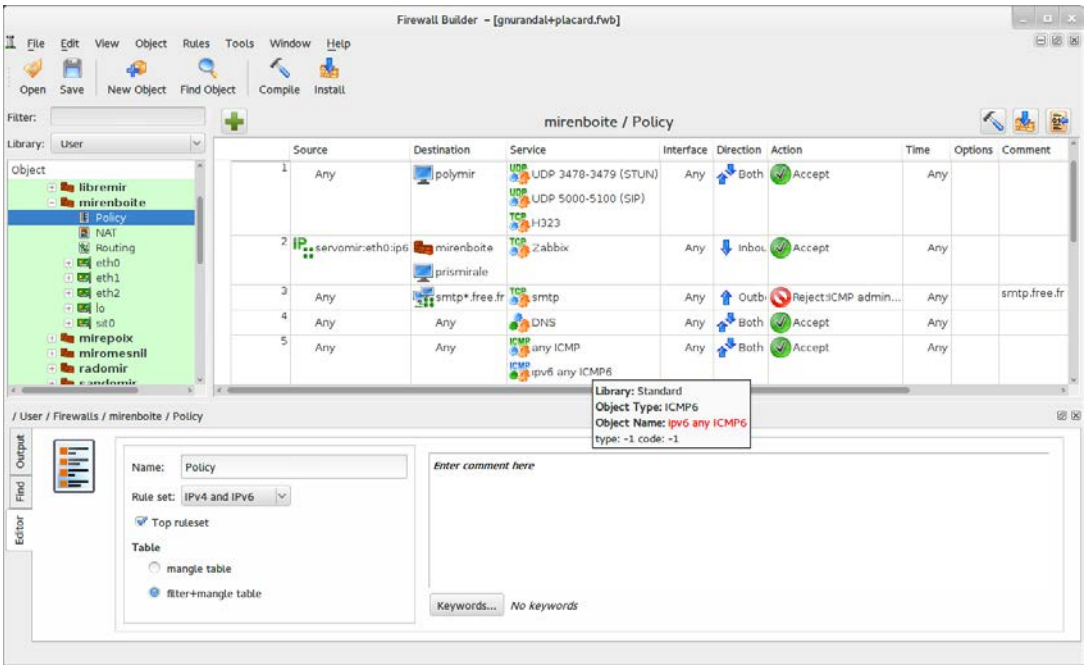


图7.2　`fwbuilder`的主窗口

`fwbuilder` 将根据你定义的规则，来生成配置防火墙的脚本。其模块化架构使其能够针对不同系统生成不同配置脚本，包括 Linux 上的 `iptables`、FreeBSD 上的 `ipf` 和 OpenBSD 上的 `pf`。

7.4.4　每次启动时加载的规则

为了每次启动主机时可以正常配置防火墙规则，需要将配置脚本注册在`/etc/network/interfaces` 文件的 `up` 目录中。在后面的例子中，脚本被存储在`/usr/local/etc/arrakis.fw` 中。

```
auto eth0
iface eth0 inet static
    address 192.168.0.1
    network 192.168.0.0
    netmask 255.255.255.0
    broadcast 192.168.0.255
    up /usr/local/etc/arrakis.fw
```

在此示例中，假定你使用 `ifupdown` 配置网络接口。如果你喜欢使用其他工具（如 `NetworkManager` 或 `systemd-networkd`），可以参考它们各自的使用文档，找到在接口启动后如何执行脚本的方法。

7.5 监控和日志

数据机密性与保护是安全的一个重要方面，但确保服务的可用性也同样重要。作为管理员和网络安全从业人员，你必须确保一切都按预期工作，你有责任检测异常行为并对服务进行及时维护。监控软件和日志软件在了解安全信息、掌控系统状态和网络行为检测方面起着关键作用。本节我们将介绍一些可应用于 Kali 系统监控和日志记录的几个工具。

7.5.1 使用logcheck来监控日志

`logcheck` 程序默认每小时都会检查记录一次监控日志，并通过电子邮件发送异常日志消息给管理员做进一步分析。

需要被监控的文件列表存储在`/etc/logcheck/logcheck.log` 文件中。软件的默认配置文件是`/etc/rsyslog.conf`，如果没有特殊设置，可以直接应用该配置运行。

`logcheck` 可以针对不同需求设定不同报告级别：偏执级别、服务器级别和工作站级别。偏执级别的日志记录是非常详细的，应该应用到像防火墙这样有特别需求的服务器上。服务器级别是默认级别，建议大多数服务器使用该级别。工作站显然是为工作站设计的级别，并且统计的日志也非常简单，相比其他级别过滤掉了很多信息。

在这三种情况下，`logcheck` 需要被定制以排除某些信息（取决于安装的服务），除非管理员真想每小时都收到一批没有太多价值的邮件。由于信息筛选机制相当复杂，碰到问题就需要自己翻阅 `/usr/share/doc/logcheck-database/README.logcheck-database.gz`。

适用的规则可以分成以下几种类型：

- 被认为是尝试侵入的信息（文件存储在 /etc/logcheck/cracking.d/ 目录中）。

- 被忽略的尝试入侵事件（/etc/logcheck/cracking.ignore.d/）。
- 被标识为安全警报信息（/etc/logcheck/violations.d/）。
- 被忽略的安全警报事件（/etc/logcheck/violations.ignore.d/）。
- 最后是其余信息（可以视为系统事件）。

ignore.d 文件(显然)用于忽略消息。例如，任何被标识为侵入尝试和安全警报的信息（按照 `/etc/logcheck/violations.d/myfile` 文件中的规则）只能由`/etc/logcheck/violations.ignore. d/myfile`，或者 `/etc/logcheck/violations.ignore.d/myfile-extension` 文件中定义的规则忽略。

系统事件总是会被发送，除非在`/etc/logcheck/ignore.d.{paranoid，server，workstation}/` 目录中的规则指明某些事件要被忽略。当然，只有相应详细等级等于或大于所选择模式的目录才会起作用。

7.5.2　监控实时行为

`top` 是一个可以显示当前运行进程的交互工具。默认排序基于处理器的使用量，可以通过 P 键获取。其他排序方法包括内存使用量（M 键）、总处理器时间（T 键）和处理器标识（N 键）。k 键允许输入进程标识结束进程，而 r 键改变进程的优先级。

当系统看起来超载的时候，`top` 是一个很好的工具，它可以用来查看哪个进程抢占处理器或者消耗过多内存。特别是，检查那些进程所消耗的资源是否匹配宿主机器服务要使用的。以“www-data”用户身份运行的未知进程应该被列入重点排查对象，它很有可能是通过 Web 应用上的漏洞，在系统上安装并运行的。

`top` 是一个很灵活的工具，其手册页中详细列出了如何定制显示，以适合个人需求和习惯。

`gnome-system-monitor` 是一个和 `top` 类似的图形工具，其提供大体上相同的功能和特性。

7.5.3　侦测变化

系统一旦安装和配置完成后，除非安全更新，通常不会修改除数据文件外的大部分文件和目录。因此，留意文件有没有被人修改过，是发现入侵的一个不错的办法。任何非预期的变更都应该引起我们的警觉并对其进行进一步探查。本节介绍几个可以监视文件、检测文件变化并在发生未预期变更时通知管理员的工具。

使用dpkg --verify审计软件包

dpkg --verify（或 dpkg -V）是一个很好的工具，因为它会显示系统中被（潜在被入侵）修改的文件，但是这个输出应该是一个加盐的结果。为了完成校验工作，dpkg 需要依靠存储在硬盘数据库的校验和（可在/var/lib/dpkg/info/package.md5sums 中找到）。一个周密的攻击者会注意修改这些文件，使其包含被破坏文件新的校验和，一个高级的攻击者甚至会对你选择的 Debian 镜像源上的软件包做手脚。为防止此类攻击，我们可以使用 APT 的数字签名验证系统（见 8.3.6 节内容）来验证包。

什么是文件指纹

提醒一下，指纹是一个值，通常是数字（即使以十六进制表示），它包含一种文件内容的签名。这个签名是用一种算法（MD5 或 SHA1）计算得出的，它保证了即使文件内容发生微小的变化，其指纹也会发生明显变化，这被称为“雪崩效应”。这可以让我们以一个简单的数字指纹作为基础来检查一个文件内容是否被修改。这类算法是不可逆的，换句话说，对于大多数人来说，即便知道指纹也无法得知相应的文件内容。虽然最近数学上的进步似乎削弱了这些原理的绝对性，但是到目前为止它们的可用性并没有受到质疑，因为制造产生相同指纹的两个不同内容仍然是一项相当困难的任务。

运行 dpkg –V 命令将验证所有已安装的软件包，并将为未通过验证的每个文件输出一行。每个字符表示对某些特定元数据的测试。不幸的是，dpkg 现阶段不存储大多数测试所需的元数据，因此会输出问号。目前仅支持校验测试，当校验失败时在第三个字符位置上显示字符 5。

```
# dpkg -V
??5??????  /lib/systemd/system/ssh.service
??5?????? c /etc/libvirt/qemu/networks/default.xml
??5?????? c /etc/lvm/lvm.conf
??5?????? c /etc/salt/roster
```

在上面的例子中，dpkg 报告了管理员对 SSH 服务软件包文件的一次不合法修改，恰当的做法应该是使用/etc/systemd/system/ssh.service 配置文件（所有配置文件的修改都应该存储在/etc 目录下面）。它也会列出多个经过合法修改的配置文件（在第二个字段中用字母“c”进行标识）。

监控文件：AIDE

AIDE（Advanced Intrusion Detection Environment ）工具可以检查文件完整性，侦测系统之前文件镜像的任何变化。这种数据被储存在数据库（/var/lib/aide/aide.db）中，包

含了系统上所有文件的相关信息（指纹、权限、时间戳等）。该数据库可以用 `aideinit` 命令进行初始化，然后每天（通过 `/etc/cron.daily/aide` 脚本）检查有无相关改变。如果探测到变化，AIDE 会将其记入文件（`/var/log/aide/*.log`）并将发现通过邮件发送给管理员。

保护数据库	由于 AIDE 使用本地数据库来比较文件的状态，因此数据库的有效性直接影响结果。如果攻击者获得超级用户权限，他就能替换数据库并覆盖其痕迹。一个可能的替代方案就是将参考数据存储在只读媒介上。

`/etc/default/aide` 中的很多选项都可以用于调整 AIDE 软件包的行为。AIDE 配置存储在`/etc/aide/aide.conf` 和`/etc/aide/aide.conf.d/`文件中（实际上，这些文件只用于 `update-aide.conf` 生成`/var/lib/aide/aide.conf.autogenerated`）。配置指明哪些文件的哪些特性需要检查。例如，日志文件的内容是经常变化的，只要这些文件的读写权限保持不变，那么就可以忽略其他变化。但是，可执行程序的内容和权限必须是不变的。虽然不是很复杂，但是配置语法不是很直观，推荐大家阅读 `aide.conf(5)` 手册。

新版本的数据库每天都会在 `/var/lib/aide/aide.db.new` 中生成，如果所有记录的变化都是合法的，就可以用它替换参考数据库。

Tripwire 和 AIDE 非常相似，甚至配置文件的语法都几乎一样。Tripwire 所提供的额外特性是对配置文件进行签名的机制，这样攻击者就不能将其指向参考数据库中的不同版本。

Samhain 也提供了类似的功能，并能帮助检测 rootkit（参阅下面的“checksecurity 和 chkrootkit / rkhunter 软件”的介绍）。它也可以在网络上被全局部署，并且其跟踪记录的结果（带签名）被保存在中央服务器上。

checksecurity 和 chkrootkit/ rkhunter 软件	checksecurity 包含的几个简单脚本可以为系统做一些安全检查（搜索空白口令、新的 setuid 文件等），并会在检查发现问题时通知管理员。尽管其名为“安全检查”，但是我们不能仅仅指望它来确保 Linux 系统的安全。 chkrootkit 和 rkhunter 软件可以搜索系统上安装的潜在恶意软件（rootkits）。提示大家一下，此类恶意软件被设计成以非常隐蔽的方式潜藏在系统中，并背地里控制机器。检测并不能保证 100%可靠，但是可以告诉我们这些潜在问题非常值得重视。

7.6 小结

在本章中，我们介绍了安全防护策略的概念，重点介绍了作为一个安全从业人员在制定安全策略时，应该重点考虑实际需求和面对的威胁，从而制定相应的解决方案。我们也讨论了笔记本电脑和桌面电脑的防火墙和包过滤等安全设置措施。最后，介绍了监控工具和相关的安全策略，并告诉大家如何使用它们来检测系统的潜在威胁。

要点提示：

- 不妨多花点时间去设计相对全面的安全策略。
- 如果用 Kali 作为一个公网可访问服务器的操作系统，在启动服务和系统正式上线之前，要对其中的所有含有默认密码配置的服务（见 7.3 节）进行修改，并通过防火墙策略限制对它们的访问（参阅 7.4 节）。
- 可以使用 `fail2ban` 检测并防止本地密码猜解和远程暴力破解密码攻击。
- 如果运行 Web 服务，可以通过 HTTPS 加密，以防止被网络中其他人通过嗅探得到重要信息（可能包括身份验证 Cookie）。
- 很多时候风险发生在你把设备从一个地方转移到另一个地方的过程中。例如，你的笔记本电脑可能在旅行途中被盗，或被海关扣留。通过使用全磁盘加密可以有效防止因丢失设备带来的进一步后果（参见 4.2.2 节），并考虑使用 nuke 特性来保护你的客户数据（参阅 9.4.4 节）。
- 部署防火墙规则（参阅 7.4 节）可以禁止除了指定 VPN 链路对外流量之外的所有出站流量。这是为了保证你的网络处于安全状态，当 VPN 链接断开时，你立即能够知道（而不是回到本地网络进行外部访问）。
- 禁用不使用的服务。Kali 为了减轻用户的工作，已经把所有可被网络访问的服务默认设置成禁用状态了。
- Linux 内核中嵌入了 netfilter 防火墙。但因为根据网络和用户需求不同，防火墙的策略差异也非常大，所以没有为我们设置任何默认策略。我们可以利用 `iptables` 和 `ip6tables` 命令从用户空间来设置防火墙规则。
- `logcheck` 程序默认每小时巡查一次并生成日志文件，并且将异常信息通过邮箱发送给管理员，以便做进一步的分析。`top` 是一个交互式工具，可以显示当前正在运行的进程列表。
- `dpkg --verify`（或 `dpkg -V`）依靠校验和作为参考，找到那些被修改过的系统文件（可能被攻击者修改），但是一个聪明的攻击者可能会破坏校验和以掩盖攻击痕迹。

- AIDE（Advanced Intrusion Detection Environment ）工具可以检查文件的完整性，侦测系统文件镜像发生的任何变化。
- Tripwire 和 AIDE 非常相似，甚至配置文件的语法都几乎一样。Tripwire 所提供的额外特性是对配置文件进行签名的机制，这样攻击者就不能将其指向参考数据库中的不同版本。
- 在很多情况下可以考虑使用 `rkhunter`，它可以对系统进行安全性检查，`chkrootkit` 可以用来帮助检测系统中是否存在 rootkit（隐蔽恶意软件）。

在下一章中，我们将讨论 Debian 的基本配置和安装包管理。很快你就会了解到作为 Kali 底层基础的 Debian 所拥有的强大能力，并了解开发人员如何使用这种强大能力。要注意的是，下一章的内容相当复杂，但它们都至关重要，如果你打算成为一名专业的 Kali 用户，那么你必须了解 Debian 的基本配置和软件包管理。

练习题

练习1——加固Kali网络

1．确定你的 Kali 主机开放的端口。
2．配置你的 Kali 防火墙，使其只允许对 22、80 和 443 端口上的入站 TCP 连接。
3．验证其他端口是否能够阻止其他应用程序如 netcat 的连接。
4．确保这些规则在重新启动后仍然生效。重新启动检查！

练习2——Kali服务监控

1．在你的 Kali 主机上安装 `logcheck`。

2．尝试暴力破解你自己的 SSH 服务，看看是否有日志检查到这些攻击行为，并生成攻击事件报告。

3．创建一个 logcheck 的 cron 实例，让它每小时运行一次，并在/data/$(date-time).log 目录中创建一个日志文件。

练习3——Kali文件系统安全加固

1．在你的 Kali 主机上安装 tripwire，并监视`/var/www/html/`文件夹的变更情况。

2．如果你配置得当，你会看到很多“文件系统错误”的报错提示。是你被入侵了吗？无论哪种问题，试着去修复它们。

进阶练习

练习4——无线热点

这里有一个关于 `iptables` 的很酷且有趣的用法。你可以把一台带有无线网卡的计算机通过 hostapd 变成一个无线热点。具体的解决方案如下：

```
1. iptables -t nat -F
2. iptables -F
3. iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
4. iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT
5. echo '1' > /proc/sys/net/ipv4/ip_forward
6. (DNS, dhcp still required)
```

另外，推荐你看看 reference guide for iptables 这个非常好用的参考指南。

第8章　Debian软件包管理

内容

- APT 简介
- 软件包交互基础
- 高级的 APT 配置和使用
- 软件包索引：深入 Debian 软件包管理系统
- 小结

在了解了 Linux 基础知识之后，接下来我们学习基于 Debian 发行版的软件包管理系统。在包括 Kali 在内的发行版中，Debian 软件包通过一系列标准化的流程，使软件最终可以被用户安装和使用。了解软件包管理系统将让你对 Kali 的结构有更深入的了解，使你能够更有效地排除故障，并帮助你在 Kali Linux 中快速找到关于各种工具与实用程序的帮助文档。

在本章中，我们将介绍 Debian 软件包管理系统，并介绍 dpkg 和 APT 套件工具。Kali Linux 的软件包管理系统非常灵活，这是它的一个重要特点，软件包管理工具提供了非常好用的安装、升级、删除应用软件的配置功能，甚至可以使用其去调整操作系统。了解 Kali 系统运作原理，可以帮助你更加充分地使用 Kali，并能大大简化很多烦琐的操作。需要我们进行痛苦的编译，灾难性地升级，调试 gcc、make 和 configure 问题的时代早已过去，但是应用程序数量已经爆炸性增长，而且你还得了解如何更好地利用这些程序。在这里给大家介绍一个关键的技巧，因为有许多安全工具，由于许可授权或其他问题，没有默认安装在 Kali 系统里，但它们有 Debian 版本软件包可供下载。我们非常有必要知道如何处理和安装这些软件包，以及它们会对系统产生什么样的影响，特别是在遇到一些未预期的情况时，这一点将变得尤为重要。

我们将从 APT 的基本概述开始，介绍关于二进制包和源代码包的结构与内容，讨论一些基础工具和应用场景，然后深入探索，以帮助你从这个庞大的软件包系统和工具套件中了解应该如何利用它们。

8.1 APT使用说明

我们从 dpkg 和 APT 的基本定义、概述和 Debian 软件包的历史回顾开始。

8.1.1 APT和dpkg之间的关系

Debian 软件包其实就是一个软件应用程序的压缩文件。二进制包（.deb 文件）里面包含了一些直接可用的文件（如程序或文档），而源代码包包含了软件的源代码和构建二进制包所需的指令。Debian 软件包中包含应用程序文件以及其他元数据，其中元数据包括应用程序所需的依赖应用的名称，以及在包的生命周期不同阶段可以执行的脚本命令（安装、删除和升级等）。

`dpkg` 工具通常可以很好地处理和安装各种`.deb` 软件包，但是如果遇到缺少或无法满足软件的依赖关系（如缺少库）时，它将阻止软件包的安装，因为它无法根据逻辑来处理这些依赖关系，`dpkg` 只能简单地列出缺失的依赖项。相比而言，高级打包工具（APT），包括 `apt` 和 `apt-get`，则可以很智能地处理并自动解决类似问题。我们将在本章下面部分讨论 `dpkg` 和 APT 工具。

`dpkg` 命令是系统上用于处理 Debian 软件包的基本命令，它用来解压、分析和安装.deb 软件包及包内的其他内容。然而 `dpkg` 只有关于 Debian 的一个局部系统视图：它知道系统上安装了什么，以及你在命令行上输入了什么，但是对其他可用软件包一无所知。因此，如果现有环境没有满足软件的依赖关系，此次安装则宣告失败。好在 APT 解决了这个限制。

APT 是一套协助管理 Debian 软件包和 Debian 系统上安装应用程序的工具。你可以使用 APT 来安装和删除应用程序、更新软件包以及升级系统。APT 的神奇之处在于它是一个完整的安装包管理系统，它不但可以用来安装或移除一个软件包，还会考虑应用程序安装的需求和依赖关系（甚至依赖应用所需求的依赖关系），并自动安装为满足安装所需的各种条件。APT 依赖于 `dpkg`，但与 `dpkg` 不同，因为前者会从一个在线源安装最新版本包，并努力解决依赖关系问题，而 `dpkg` 只能安装一个位于本地系统上的文件包，而且不会自动解决依赖关系。

如果你用过 Linux 系统很长的时间，应该会有使用 gcc 来编译程序（甚至在 `make` 和 `configure` 等工具的帮助下）的经验，那么你肯定会记得那是一个痛苦的过程，特别是在应用程序有多个依赖关系时，这个过程尤为复杂。你要通过编译过程中的各种警告和错误消息，试着确定代码的哪一部分导致报错，并且推断出该异常是由于缺少哪个库，还是需要其

他哪些依赖项导致的。而后你再去寻找缺少的库或依赖项，再试着安装一次。那么，如果你幸运的话，编译顺利完成，但更大的可能是编译再次失败，提示需要另一个依赖包。

APT 的出现就是为了帮助我们解决这个问题的，它会帮我们整理程序对运行环境的需求和依赖关系，并协助我们解决。APT 默认可以直接在 Kali Linux 上使用，但 APT 并不是万能的。我们还是需要了解 Debian 和 Kali 的软件包系统是如何工作的，这对你安装软件包、更新软件或解决软件包的各种问题都是非常有帮助的。APT 会贯穿你使用 Kali Linux 的日常工作中，在本章中，我们将介绍 APT 这个工具，并告诉你如何用它来安装、删除、升级和管理软件包，以及告诉你如何在不同的 Linux 发行版之间转移使用软件包。我们也会讨论基于 APT 的图形化工具，展示如何验证软件包来源的真实性，并深入研究“滚动升级”这一概念，此项技术用于日常更新 Kali Linux 系统。

在深入介绍如何使用 dpkg 和 APT 来安装与管理软件包之前，我们先研究一下 APT 的工作原理，并讨论一些和它有关的术语。

软件源和源代码包　“源”这个词的意思可能会有一些模糊不清，我们在这里简单介绍一下和它有关的一些概念。源代码包是指一个包含有源程序代码的包，不要和“软件源”这一概念相混淆，软件源是指一个软件包所在的软件库（位于网站、FTP 服务器、CD-ROM 光驱或本地磁盘等）。

APT 从软件库（软件包存储系统，或简单地称为软件源）来获取和数据包有关的数据。`/etc/apt/sources.list` 文件中包含了 Debian 软件包的公开软件库（源）。

8.1.2　理解sources.list文件

`sources.list` 文件是定义软件源的关键配置文件，了解它的结构以及如何配置非常重要，因为 APT 无法在没有正确定义软件源列表的情况下运行。我们来看看它的语法，看看 Kali Linux 能使用的各种软件库、镜像和重定向资源，然后就可以来使用 APT 了。

`/etc/apt/sources.list` 文件（以及`/etc/apt/sources.list.d/*`列表文件）的每一行都包含了一个源的描述，由三部分组成，由空格分隔。注释部分由#符号开头：

```
# deb cdrom:[Debian GNU/Linux 2016.1 _Kali-rolling_ - Official Snapshot
    ➥ amd64 LIVE/INSTALL Binary 20160830-11:29]/ kali-rolling contrib main
    ➥ non-free

deb http://http.kali.org/kali kali-rolling main non-free contrib
```

我们来看看这个文件的语法。第一个字段表示软件源类型：

- deb 表示二进制包
- deb-src 表示源代码包

第二个字段是软件源的 URL 地址。它可以包含 Debian 镜像或由第三方部署的其他软件包。该网址可以以 `file://`开头，表示在系统的文件层次结构中安装的本地软件源，也可以用 `http://`表示可从 Web 服务器访问的软件源，或以 `ftp://`开头表示部署在 FTP 服务器上可用的软件源。URL 也可以以 `cdrom` 开头，表示基于 CD-ROM / DVD-ROM / Blu-raydisc 的软件源，但由于基于网络的安装方法越来越普遍，所以这种源使用的频率较低。

在 `cdrom` 这个条目中需要添加具体使用的 CD-ROM / DVD-ROM。与其他类型条目不同，CD-ROM 并不总是可用的，因为必须有光盘插入驱动器中，并且通常一次只能读取一个盘片。由于这些原因，这种源的管理方式和网络源稍有不同，需要安装 `apt-cdrom` 程序，通常使用 `add` 参数执行，然后将会弹出请求，提示将光盘插入驱动器，放入光盘即可浏览其内容，查找需要的软件包文件。我们也可以使用光盘中的这些文件，来更新可用包的数据库列表（这个操作通常通过 `apt update` 命令完成）。之后，APT 将会查询光盘是否存在所需的安装包。

最后一个字段的内容取决于软件库的存储结构。最简单的情况是，它直接指向所需软件源的子目录（通过在后面添加 / 和相应的路径可指向下一级目录，还有一种情况就是结尾处只有一个简单的“./”，它指的是当前目录下没有子目录了，所有包直接在指定 URL 目录上）。但在最常见的情况下，软件库的结构就像一个 Debian 镜像，具有多个发行版的目录，每个发行版又具有多个组件。在这些情况下，根据命名来选择发行版，然后再启用想要的组件（或 section）。下面我们具体来介绍。

Debian 和 Kali 根据每个软件作品开发者所选择的许可证，把软件包分成三类。

- main类型的软件包是完全遵从Debian Free Software Guidelines[1]的软件包。
- non-free（非开源）类型则不同，因为它不完全符合开源许可，但它可以不受限制地被分发。
- Contrib（贡献）类型是指那些必须依赖 `non-free` 元素运行的一些开源软件。这些元素可能是 `non-free` 部分的软件，或非开源文件如游戏 ROM、BIOS 控制器驱动等。Contrib 类型还包括那些需要由作者提供专有元素才能使用的免费软件，如 VirtualBox，它需要一个非开源的编译器来构建一些文件才能运行。

现在，我们来看看那些标准的 Kali Linux 软件源或软件库。

1 https://www.debian.org/social_contract#guidelines

8.1.3　Kali软件库

Kali Linux 系统的标准 `sources.list` 文件已经包含了一个软件库（kali-rolling）和前面提到的三类组件：`main`、`contrib` 和 `non-free`。

```
# Main Kali repository
deb http://http.kali.org/kali kali-rolling main contrib non-free
```

我们来看看各种 Kali 软件库。

Kali-Rolling 库

这是用户的主要存储库。它包含了可安装的最新软件包。它通过一个工具对 Debian 测试版和 Kali 特定软件包进行合并操作，来确保在 Kali-Rolling 中每个包的依赖关系都可以得到满足。换句话说，除非管理器脚本中有什么错误，否则所有的软件包都应该是可安装的。

因为 Debian 测试版每天都在发展进化，Kali-Rolling 也是如此。像其他那些重要软件会频繁更新一样，Kali 专用软件包也会定期更新。

kali-Dev库

这个存储库不是供公众使用的。在这里，Kali 开发人员会解决 Kali 特定软件包合并到 Debian 测试版中引起的冲突或依赖问题。

这也是更新软件包最先着陆的地方，所以如果你需要一个还没有到达 Kali-Rolling 的更新，你可以从这个库中获取它。但这对普通用户而言是不建议的。

Kali-Bleeding-Edge库

这个库自动包含了上游 Git（或 Subversion）库中的软件包。这样做的好处是你可以访问软件的最新功能和 24 小时以内提交的错误修复补丁。这是验证你向上游提交问题报告是否已被修复的理想途径。

缺点是这些软件包还没有经过测试或核查。如果上游发生变化影响了软件包（比如增加了一个新的依赖），那么这个软件包可能无法使用。正因为这个原因，APT 会对它进行标记，不会自动从它安装软件包，特别是在升级时。

你可以通过编辑`/etc/apt/sources.list` 来注册软件库，或是在`/etc/apt/sources.list.d` 目录下重新创建一个文件，后一种方法可以使得原来的系统 `sources.list` 文件保持未被更改。在这个例子中，我们选择创建一个单独的`/etc/apt/sources.list.d/kali-`

bleeding-edge.list 文件，内容如下：

```
# Kali Bleeding Edge repository
deb http://http.kali.org/kali kali-bleeding-edge main contrib non-free
```

Kali Linux镜像

上面的sources.list指向http.kali.org。这是一个运行MirrorBrain[1]的服务器，其将把你的HTTP请求重定向到离你较近的官方镜像。MirrorBrain监视每个镜像服务，以确保它们正常运行，并且是最新的版本。它会把你重定向到一个运行状态良好的镜像站点。

调试镜像重定向

如果你的镜像源出了问题（例如运行 apt updage 命令失败了），则可以使用 curl –sI 命令来看看什么地方被重定向了：

```
$ curl -sI http://http.kali.org/README
HTTP/1.1 302 Found
Date: Mon, 11 Apr 2016 09:43:21 GMT
Server: Apache/2.4.10 (Debian)
X-MirrorBrain-Mirror: ftp.free.fr
X-MirrorBrain-Realm: country
Link: <http://http.kali.org/README.meta4>; rel=describedby;
    ➥ type="application/metalink4+xml"
Link: <http://ftp.free.fr/pub/kali/README>; rel=duplicate;
    ➥ pri=1; geo=fr
Link: <http://de-rien.fr/kali/README>; rel=duplicate;
    ➥ pri=2; geo=fr
Link: <http://ftp.halifax.rwth-aachen.de/kali/README>;
    ➥ rel=duplicate; pri=3; geo=de
Link: <http://ftp.belnet.be/kali/kali/README>;
    ➥ rel=duplicate; pri=4; geo=be
Link: <http://ftp2.nluug.nl/os/Linux/distr/kali/README>;
    ➥ rel=duplicate; pri=5; geo=nl
Location: http://ftp.free.fr/pub/kali/README
Content-Type: text/html; charset=iso-8859-1
```

如果这个问题依然存在，可以编辑/etc/apt/sources.list，然后硬编码一个运行状态正常的镜像网站，来替代 http.kali.org 这条记录。

还有第二个 MirrorBrain 实例：http.kali.org 维护软件包存储库，cdimage.kali.org 站点上存

1 http://mirrorbrain.org

储了已发布的 ISO 镜像。

- `http://cdimage.kali.org`

如果你想要请求正式的 Kali Linux 镜像列表，则可以将.mirrorlist 添加到任何有效的 URL 中，例如指向 http.kali.org 或 cdimage.kali.org 的后面。

- `http://http.kali.org/README.mirrorlist`
- `http://cdimage.kali.org/README.mirrorlist`

由于 MirrorBrain 存在的一些限制，这些列表并不详尽（受限于某些国家网络访问的限制，很多好用的镜像并不会出现在列表中，除非你在这些国家）。但是其中包含了最好的镜像，并且它们运行良好，提供了大量稳定的可用带宽。

8.2　软件包基础交互

在基本了解了 APT 之后，我们来看看一些 APT 软件包的基础操作，包括 APT 初始化、安装、卸载和清理以及 Kali Linux 系统升级。然后我们走出命令行，看看图形化的 APT 工具。

8.2.1　APT初始化

APT 是一个庞大的项目和工具集，并在最初设计时就考虑并预留了一个图形化接口。从用户的角度来看，它是一个以 `apt-get` 和 `apt` 为核心命令行方式运行的工具。

实际上有多种第三方图形化工具，可供用户选择使用，例如 `synaptic` 和 `aptitude`，我们将在后面讨论。尽管这里更倾向于推荐大家使用 `apt` 这个工具，但我们还是会详细介绍其他的一些工具，以及它们之间的一些主要语法差异。

在使用 APT 时，你应该首先使用 `apt update` 来下载当前可用软件包的列表。因为各种软件包列表、源代码列表和翻译文件的规模都随着 Debian 的发展不断扩大，根据网络连接速度的不同，这可能需要一些时间。当然，如果使用 CD-ROM/ DVD-ROM 来更新或安装，那肯定要快得多，因为这是直接从本地安装。

8.2.2　安装软件包

由于 Debian 软件包系统在设计时考虑得很周全，所以我们可以不用去考虑它的依赖关

系，它的安装很容易。下面我们来看看如何使用 dpkg 和 apt 安装软件包！

使用dpkg安装软件包

当你需要安装软件包时，dpkg 是你的核心工具（直接使用或通过 APT 间接使用）。即便是离线使用，它也是首选，因为它本身就不需要 Internet 连接。注意，dpkg 不会帮你安装该软件包可能需要的任何依赖项。使用 dpkg 安装软件包之前，你必须已经下载了（或以其他方式获得）要安装软件包的.deb 文件，然后只需提供-i 或--install 安装选项以及.deb 的路径，就可以开始安装软件了。

```
# dpkg -i man-db_2.7.0.2-5_amd64.deb
(Reading database ... 86425 files and directories currently installed.)
Preparing to unpack man-db_2.7.0.2-5_amd64.deb ...
Unpacking man-db (2.7.0.2-5) over (2.7.0.2-4) ...
Setting up man-db (2.7.0.2-5) ...
Updating database of manual pages ...
Processing triggers for mime-support (3.58) ...
```

我们可以清楚地看到，dpkg 执行了哪些步骤，并可以看到在任何时间点发生的异常情况。-i 或--install 选项将自动执行两个步骤：解压缩软件包和运行配置脚本。你也可以分别使用-unpack 和--configure 选项独立执行这两个步骤（与 APT 在后台运行一样），来进行安装：

```
# dpkg --unpack man-db_2.7.0.2-5_amd64.deb
(Reading database ... 86425 files and directories currently installed.)
Preparing to unpack man-db_2.7.0.2-5_amd64.deb ...
Unpacking man-db (2.7.0.2-5) over (2.7.0.2-5) ...
Processing triggers for mime-support (3.58) ...
# dpkg --configure man-db
Setting up man-db (2.7.0.2-5) ...
Updating database of manual pages ...
```

请注意，在上面的例子中，“Processing triggers”这一行指的是在被监视目录中发生添加删除或者修改文件操作时自动执行的代码。例如，mime-support 软件包监视 /usr/lib/mime/packages 目录，每当这个目录中有任何变化，就执行 update-mime 命令（比如/usr/lib/mime/packages/man-db 目录中 man-db 的变化）。

当 dpkg 无法安装某个软件包时系统会返回错误信息。尽管如此，我们还是可以试着让 dpkg 忽略这个错误，只需要添加--force-*选项来跳过这个问题，就可以让 dpkg 强行安装软件包。我们可以使用 dpkg --force-help 命令来显示这个选项的完整参数列表。例

如，你可以使用 dpkg 来强行安装 zsh：

```
$ dpkg -i --force-overwrite zsh_5.2-5+b1_amd64.deb
```

在实际的使用过程中，你可能会遇到的一个很常见的错误就是文件冲突。当一个软件包需要安装一个已经由另一软件包安装过的文件时，dpkg 会拒绝安装这个软件包。然后显示如下所示的错误提示：

```
Unpacking libgdm (from .../libgdm_3.8.3-2_amd64.deb) ...
dpkg: error processing /var/cache/apt/archives/libgdm_3.8.3-2_amd64.deb (--
    ➥ unpack): trying to overwrite '/usr/bin/gdmflexiserver', which is also
    ➥ in package gdm3 3.4.1-9
```

在这种情况下，如果你知道替换这个文件对系统的稳定性没有多大影响（通常是这种情况），则可以使用--force-overwirte 命令覆盖该文件进行安装。

虽然--force-*有许多可用的选项，但可能只有--force-overwirte 是最常用的。其他的选项是针对特殊情况而存在的，为了遵守软件包机制所规定的规则，最好不要使用这些参数。别忘了，这些规则要有效确保系统的一致性和稳定性。

使用APT安装包

APT 比 dpkg 先进得多，并且它在后台帮你做了很多工作，你会发现使用 APT 与软件包交互很简单。你可以使用 apt install package 这样一个简单的命令，向系统添加一个简单的安装包。APT 还能自动帮你安装那些必要的依赖项：

```
# apt install kali-linux-gpu
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  oclgausscrack oclhashcat
The following NEW packages will be installed:
  kali-linux-gpu oclgausscrack oclhashcat
0 upgraded, 3 newly installed, 0 to remove and 416 not upgraded.
Need to get 2,494 kB of archives.
After this operation, 51.5 MB of additional disk space will be used.
Do you want to continue? [Y/n]
Get:1 http://archive-2.kali.org/kali kali-rolling/non-free amd64 oclhashcat
    ➥ amd64 2.01+git20160114-0kali2 [2,451 kB]
Get:2 http://archive-2.kali.org/kali kali-rolling/main amd64 oclgausscrack
    ➥ amd64 1.3-1kali2 [37.2 kB]
Get:3 http://archive-2.kali.org/kali kali-rolling/main amd64 kali-linux-gpu
```

```
    ➥ amd64 2016.3.2 [6,412 B]
Fetched 2,494 kB in 0s (3,060 kB/s)
Selecting previously unselected package oclhashcat.
(Reading database ... 317084 files and directories currently installed.)
Preparing to unpack .../0-oclhashcat_2.01+git20160114-0kali2_amd64.deb ...
Unpacking oclhashcat (2.01+git20160114-0kali2) ...
Selecting previously unselected package oclgausscrack.
Preparing to unpack .../1-oclgausscrack_1.3-1kali2_amd64.deb ...
Unpacking oclgausscrack (1.3-1kali2) ...
Selecting previously unselected package kali-linux-gpu.
Preparing to unpack .../2-kali-linux-gpu_2016.3.2_amd64.deb ...
Unpacking kali-linux-gpu (2016.3.2) ...
Setting up oclhashcat (2.01+git20160114-0kali2) ...
Setting up oclgausscrack (1.3-1kali2) ...
Setting up kali-linux-gpu (2016.3.2) ...
```

你也可以使用 `apt-get install` package，或者 `aptitude install` package 命令，来安装软件包。在安装软件包这件事情上，这几个工具做的事是一样的。在后面介绍中，你会看到，在重大升级或者处理复杂的依赖关系时，这些工具的特点就会凸显出来。

如果 `sources.list` 列出了几个发行版本，你可以使用 `apt install package=version` 来指定你想要的软件包版本，但是通常使用 `apt install package/distribution` 指定从哪个源（kali-rolling、kali-dev 及 kali-bleeding-edge）来更新是一个更好的选择。

和 `dpkg` 一样，你也可以使用 APT 强行安装一个软件包并使用`--force-overwrite` 覆盖文件，但是这个语法有点奇怪，因为它会把参数传递给 `dpkg`。

```
# apt -o Dpkg::Options::="--force-overwrite" install zsh
```

8.2.3 更新Kali Linux

作为一个滚动升级版的 Linux 系统，Kali Linux 具有强大的升级能力。在这一节中，我们将会看到升级 Kali 是一件多么简单的事情，我们还将讨论如何制定你的更新策略。

建议定期对系统进行升级，因为通过升级可以为系统安装最新的安全更新补丁。升级的过程也十分简单，先用 `apt update` 命令更新软件源中的软件列表，然后用 `apt upgrade`、`apt-get upgrade` 或 `aptitude safe-upgrade` 命令来更新已安装的软件。这些命令只会对那些可以被升级的软件起作用，而不会删除你的任何软件包。也就是说通过运行这些命令可以在保证完成更新软件的前提下尽可能不带来任何麻烦。在更新系统上，

apt-get 命令的使用略苛刻于 aptitude 和 apt 命令，因为它会拒绝安装未预先安装的软件包。

apt 工具通常会选择安装软件的最新版本（不包括 kali-bleeding-edge 源中软件包的最新版本，默认情况下，此源的软件版本号是被忽略的）。

如果想从特定的软件库搜索或升级软件包，可以使用-t 或--target-release 参数，再跟所需软件库的名称（例如，apt -t kali-rolling upgrade）。为了避免每次使用 apt 时都需要指定这个选项，可以在文件/etc/apt/apt.conf.d/local 中添加 apt :: Default-Release “kali-rolling”。

对于更重要的升级，例如系统主版本升级，可以使用 apt full-upgrade 命令完成升级。apt 指令将自动完成所有与系统升级有关的工作，包括必要的旧版本软件卸载，或者安装新的依赖项等操作。这个命令也适用于定期升级 Kali 系统。这是不是很简单？几乎不需要什么特别的操作。APT 正是基于这个伟大的设计而名声鹊起的。

与 apt 和 aptitude 不同，apt-get 并没有 full-upgrade 命令。你可以使用 apt-get dist-upgrade（distribution upgrade）命令来代替 full-upgrade 命令，并且 apt 和 aptitude 出于向后兼容性考虑也支持它。

预见重要的改变　为了能够预见一些重要问题，你可以安装 apt-listchanges 软件包，这个软件可以告诉你更新一个软件包时可能遇到的各种问题。这些信息是由软件包维护人员编写的，并放在 /usr/share/doc/package/NEWS.Debian 文件中。阅读这些文件（最好通过 apt-listchanges 来查看），可以有效帮你避免很多麻烦。

自从 Kali 变为滚动发行之后，甚至一天可以升级好几次。然而，一天升级好几次可不是什么好事。那么，我们应该多久升级一次 Kali Linux 呢？其实这并没有一个硬性的标准，但有一些指导性建议，你可以参考。可以按照以下几点建议来升级系统：

- 当你意识到在更新中修复安全问题时。
- 怀疑更新后的版本可能会修复你遇到的错误。
- 在提交错误报告之前，确保你的系统是最新版本，并且问题可以复现。
- 通常升级系统可以获得一些你没有听说过的安全修复程序。

也有一些情况，最好不要升级。例如，以下情况可能就不是一个升级系统的好时候：

- 如果你正在做的工作无法承受任何中断（例如，电脑可能即将断网，或者你稍后要用电脑来做一个展示报告），最好等以后有足够时间来解决升级过程中可能引入的任何问题时，再来升级系统。
- 如果最近有一个重大的版本升级，你担心新版本的系统可能还有问题没有被发现。例

如，当发布一个新的 GNOME 版本时，通常不是全部软件包一次性更新，很可能是在软件包新旧版本共同使用的过渡期。大多数情况下不会出现什么问题，它可以帮助大家逐步完成这些更新，但总是有例外，一些应用程序可能由于这种差异而被破坏，导致无法正常工作。

- 如果 `apt full-up grade` 命令的输出提示将删除某个对你的工作很重要的软件包，在这些情况下，你需要查看具体情况，并试图弄明白为什么要移动或删除它们。也许这些软件包目前已经坏掉了，你也可能想等到固定版本可用以后，或者等到授权过期再进行更新，你需要找到这类软件的替代品以后，再以某种方式进行升级。

一般来说，建议你每周至少升级一次 Kali。你当然可以每天都进行一次升级，但这样做或者比这更频繁的更新是没有意义的。即使镜像每天同步四次，来自 Debian 的更新通常每天只有一次。

8.2.4 卸载并清除软件包

卸载软件包比安装更简单。让我们来看看如何使用 `dpkg` 和 `apt` 命令删除一个软件包。

使用 `dpkg` 删除软件包，只需要提供`-r` 或`--remove` 选项，然后再加上软件包的名称。但是，这种删除并不彻底。软件有关的配置文件、维护脚本、日志文件（系统日志）、守护程序生成的数据（例如 LDAP 服务器目录的内容或 SQL 服务器的数据库内容 ），以及由软件处理的大多数用户数据都保持不变。这种删除可以轻松地卸载程序，稍后可以使用相同配置重新安装，而且依赖项不会被删除。

正如下面的这个例子：

```
# dpkg --remove kali-linux-gpu
(Reading database ... 317681 files and directories currently installed.)
Removing kali-linux-gpu (2016.3.2) ...
```

也可以使用 `apt remove` package 命令，从系统中删除软件包。APT 将自动删除这个软件包，以及依赖于被删除包的其他软件包。像 `dpkg` 的例子一样，配置文件和用户数据不会被删除。

通过为包名添加后缀，可以使用 `apt`（或 `apt-get` 和 `aptitude`）命令在同一个命令行上安装某些软件包，同时删除其他软件包。使用 `apt install` 命令，将“ - ”添加到要删除软件包的名称后面，添加“+”到你想要安装软件包的名称中。

下面的示例显示了两种不同的安装 package1 并删除 package2 的方法。

```
# apt install package1 package2-
[...]
# apt remove package1+ package2
[...]
```

这个方法也可以用来防止某些软件被连带安装，例如由于 Recommend 造成的连带安装（稍后讨论）。一般来说，依赖关系解决器将把这个信息作为寻找替代解决方案的重要线索。

要删除与软件包相关的所有数据，可以使用 `dpkg -P package` 或 `apt purge package` 命令清除软件包。这将完全删除软件包和所有用户数据，在使用 `apt` 工具的情况下，也会删除依赖关系。

```
# dpkg -r debian-cd
(Reading database ... 97747 files and directories currently installed.)
Removing debian-cd (3.1.17) ...
# dpkg -P debian-cd
(Reading database ... 97401 files and directories currently installed.)
Removing debian-cd (3.1.17) ...
Purging configuration files for debian-cd (3.1.17) ...
```

警告！由于这是彻底的清除，建议不要轻易执行。你将会失去一切与该软件包相关联的数据。

8.2.5　检查安装包

接下来，我们来认识一些可用于检查 Debian 软件包的工具。我们将学习查询与可视化显示软件包数据库的 `dpkg`、`apt` 和 `apt-cache` 命令。

查询dpkg数据库和检查.deb文件

我们从查询内部数据库的 `dpkg` 几个选项开始说起。dpkg 的数据库存储在`/ var / lib / dpkg` 目录的文件系统里，包含多个部分：配置脚本（`/ var / lib / dpkg / info`）、软件包安装的文件列表（`/ var / lib / dpkg / info / *.list`）以及每个已安装软件包的状态文件（`/ var / lib / dpkg / status`）。可以使用 `dpkg` 与此数据库中文件进行交互。注意，大多数选项都可以使用长参数版本（一个或多个相关单词，以双短画线开头），以及一个短参数版本（单个字母，通常是长版本中一个单词的首字母，前面是单个短画线）。这个惯例很常见，叫作 POSIX 标准。

首先，我们来看看`--listfiles package`（或`-L`）命令，该命令会列出指定软件包安装

生成的文件：

```
$ dpkg -L base-passwd
/.
/usr
/usr/sbin
/usr/sbin/update-passwd
/usr/share
/usr/share/lintian
/usr/share/lintian/overrides
/usr/share/lintian/overrides/base-passwd
/usr/share/doc-base
/usr/share/doc-base/users-and-groups
/usr/share/base-passwd
/usr/share/base-passwd/group.master
/usr/share/base-passwd/passwd.master
/usr/share/man
/usr/share/man/pl
/usr/share/man/pl/man8
/usr/share/man/pl/man8/update-passwd.8.gz
[...]
/usr/share/doc
/usr/share/doc/base-passwd
/usr/share/doc/base-passwd/users-and-groups.txt.gz
/usr/share/doc/base-passwd/changelog.gz
/usr/share/doc/base-passwd/copyright
/usr/share/doc/base-passwd/README
/usr/share/doc/base-passwd/users-and-groups.html
```

接下来，`dpkg --search file`（或者`-S`）命令找到包含此文件或者路径的安装包。例如，查找包含`/ bin / date` 的包：

```
$ dpkg -S /bin/date
coreutils: /bin/date
```

`dpkg --status package`（或`-s`）命令显示已安装软件包的重要信息。例如，要搜索 coreutils 包的重要信息：

```
$ dpkg -s coreutils
Package: coreutils
Essential: yes
Status: install ok installed
```

```
Priority: required
Section: utils
Installed-Size: 13855
Maintainer: Michael Stone <mstone@debian.org>
Architecture: amd64
Multi-Arch: foreign
Version: 8.23-3
Replaces: mktemp, realpath, timeout
Pre-Depends: libacl1 (>= 2.2.51-8), libattr1 (>= 1:2.4.46-8), libc6 (>=
    ➥ 2.17), libselinux1 (>= 2.1.13)
Conflicts: timeout
Description: GNU core utilities
This package contains the basic file, shell and text manipulation
utilities which are expected to exist on every operating system.
.
Specifically, this package includes:
arch base64 basename cat chcon chgrp chmod chown chroot cksum comm cp
csplit cut date dd df dir dircolors dirname du echo env expand expr
factor false flock fmt fold groups head hostid id install join link ln
logname ls md5sum mkdir mkfifo mknod mktemp mv nice nl nohup nproc numfmt
od paste pathchk pinky pr printenv printf ptx pwd readlink realpath rm
rmdir runcon sha*sum seq shred sleep sort split stat stty sum sync tac
tail tee test timeout touch tr true truncate tsort tty uname unexpand
uniq unlink users vdir wc who whoami yes
Homepage: http://gnu.org/software/coreutils
```

`dpkg --list`（或`-l`）命令显示系统已知的软件包列表及其安装状态。也可以在输出中使用 `grep` 来搜索特定字段，或者提供通配符（如 `b *`）来搜索与特定部分搜索字符串匹配的包，并返回符合条件包的摘要信息。例如，要显示以“`b`”开头的所有软件包的摘要列表：

```
$ dpkg -l 'b*'
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-aWait/Trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name           Version      Architecture  Description
+++-==============-============-============-=================================
ii  b43-fwcutter    1:019-3      amd64        utility for extracting Broadcom 4
ii  backdoor-facto  3.4.2-0kali1 all          Patch win32/64 binaries with shel
un  backupninja     <none>       <none>       (no description available)
un  backuppc        <none>       <none>       (no description available)
ii  baobab          3.22.1-1     amd64        GNOME disk usage analyzer
[...]
```

dpkg --contents file.deb（或-c）命令列出特定.deb 文件中的所有文件：

```
$ dpkg -c /var/cache/apt/archives/gnupg_1.4.18-6_amd64.deb
drwxr-xr-x root/root       0 2014-12-04 23:03 ./
drwxr-xr-x root/root       0 2014-12-04 23:03 ./lib/
drwxr-xr-x root/root       0 2014-12-04 23:03 ./lib/udev/
drwxr-xr-x root/root       0 2014-12-04 23:03 ./lib/udev/rules.d/
-rw-r--r-- root/root     2711 2014-12-04 23:03 ./lib/udev/rules.d/60-
gnupg.rules
drwxr-xr-x root/root       0 2014-12-04 23:03 ./usr/
drwxr-xr-x root/root       0 2014-12-04 23:03 ./usr/lib/
drwxr-xr-x root/root       0 2014-12-04 23:03 ./usr/lib/gnupg/
-rwxr-xr-x root/root    39328 2014-12-04 23:03 ./usr/lib/gnupg/gpgkeys_ldap
-rwxr-xr-x root/root    92872 2014-12-04 23:03 ./usr/lib/gnupg/gpgkeys_hkp
-rwxr-xr-x root/root    47576 2014-12-04 23:03 ./usr/lib/gnupg/gpgkeys_finger
-rwxr-xr-x root/root    84648 2014-12-04 23:03 ./usr/lib/gnupg/gpgkeys_curl
-rwxr-xr-x root/root     3499 2014-12-04 23:03 ./usr/lib/gnupg/gpgkeys_mailto
drwxr-xr-x root/root       0 2014-12-04 23:03 ./usr/bin/
-rwxr-xr-x root/root    60128 2014-12-04 23:03 ./usr/bin/gpgsplit
-rwxr-xr-x root/root  1012688 2014-12-04 23:03 ./usr/bin/gpg
[...]
```

dpkg --info file.deb（或-I）命令显示指定.deb 文件的头字段内容：

```
$ dpkg -I /var/cache/apt/archives/gnupg_1.4.18-6_amd64.deb
new debian package, version 2.0.
 size 1148362 bytes: control archive=3422 bytes.
    1264 bytes,    26 lines      control
    4521 bytes,    65 lines      md5sums
     479 bytes,    13 lines   *  postinst           #!/bin/sh
     473 bytes,    13 lines   *  preinst            #!/bin/sh
 Package: gnupg
 Version: 1.4.18-6
 Architecture: amd64
 Maintainer: Debian GnuPG-Maintainers <pkg-gnupg-maint@lists.alioth.debian.
org>
 Installed-Size: 4888
 Depends: gpgv, libbz2-1.0, libc6 (>= 2.15), libreadline6 (>= 6.0), libusb-
    ➥ 0.1-4 (>= 2:0.1.12), zlib1g (>= 1:1.1.4)
 Recommends: gnupg-curl, libldap-2.4-2 (>= 2.4.7)
 Suggests: gnupg-doc, libpcsclite1, parcimonie, xloadimage | imagemagick | eog
 Section: utils
 Priority: important
```

```
 Multi-Arch: foreign
 Homepage: http://www.gnupg.org
 Description: GNU privacy guard - a free PGP replacement
  GnuPG is GNU's tool for secure communication and data storage.
  It can be used to encrypt data and to create digital signatures.
  It includes an advanced key management facility and is compliant
  with the proposed OpenPGP Internet standard as described in RFC 4880.
[...]
```

还可以使用 dpkg 将软件包版本号与--compare-versions 选项进行比较，通常由外部程序调用，包括由 dpkg 本身执行的配置脚本。

这个选项需要设置三个参数：一个版本号、一个比较操作符和另一个版本号。各种可能的操作符是：lt（严格小于）、le（小于或等于）、eq（等于）、ne（不等于）、ge（大于或等于）和 ge（严格大于）。如果比较正确，则 dpkg 返回 0（成功）；否则返回非零值（表示失败）。如下是几个比较的例子：

```
$ dpkg --compare-versions 1.2-3 gt 1.1-4
$ echo $?
0
$ dpkg --compare-versions 1.2-3 lt 1.1-4
$ echo $?
1
$ dpkg --compare-versions 2.6.0pre3-1 lt 2.6.0-1
$ echo $?
1
```

注意，最后一次比较依然失败。因为对于 dpkg，字符串“pre”（通常表示预发布）没有特别的意义，dpkg 只是将其解释为一个字符串，在这种情况下，“2.6.0pre3-1”按字母顺序大于“2.6.0-1”。当想要表示一个软件包版本号是一个预发布版本号时，我们使用代字符“～”：

```
$ dpkg --compare-versions 2.6.0~pre3-1 lt 2.6.0-1
$ echo $?
0
```

使用apt-cache和apt查询数据库中的可用软件包

apt-cache 命令可以显示存储在 APT 管理器内部数据库中的大部分信息。这些信息是一种缓存，因为它是从 sources.list 文件不同软件源收集到的。而这个收集过程是在 apt update 操作过程中发生的。

词汇

缓存

缓存是一种临时存储系统，用于处理那些读取消耗（性能方面）相对较高而又需要频繁读取的数据。这个概念可以应用于从微处理器内核到高端存储系统等不同规模的各种情况。

就 APT 管理工具来说，它所参考的 Packages 文件是 Debian 镜像上的文件。但是，每次查询都通过网络到在线包数据库搜索是非常难以实现的。这就是为什么 APT 会在本地（`/var/lib/apt/lists/`中）存储这些文件的副本的原因，是为了搜索可以在本地完成。同样，`/var/cache/apt/archives/`路径下也包含了那些已下载软件包的缓存副本，以避免在需要重新安装时再次下载它们。

为了避免因频繁升级而占用磁盘过多，应定期对`/ var / cache / apt / archives /`目录进行整理。可以使用这两个命令：`apt clean`（或 `apt-get clean`）完全清空目录；`apt autoclean`（`apt-get autoclean`）只移除因为镜像消失而无法使用或下载的软件包。还可以使用参数 `APT :: Clean-Installed` 来防止删除当前处于安装状态的`.deb` 文件。另外，请大家注意，在安装完成后，APT 会把下载的相关文件删除掉，而删除下载文件的操作，可能是你在使用其他工具时需要非常注意的事情。

`apt-cache` 命令可以使用 `apt-cache` keyword 来执行基于关键字的软件包搜索。它还可以使用 `apt-cache show` package 命令来显示软件包可用版本的信息。这个命令提供了软件包的各种描述信息，包括依赖关系和维护者名字等。此功能在确定系统预装的软件包（如 kali-linux-wireless、kali-linux-web 和 kali-linux-gpu）信息时显得特别有用。注意，`apt search`、`apt show`、`aptitude search` 和 `aptitude show` 这些命令也是类似的。

替代工具：`axi-cache`

`apt-cache search` 是一个非常基础的工具，其可以使用 `grep` 在数据包描述上进行筛选。如果包含太多的关键字，它往往会返回太多结果，或者根本没有结果。

然而，`axi-cache search` 命令提供了一个按相关性排序的更好结果。它使用的 Xapian 搜索引擎是 apt-xapian-index 软件包的一部分，它会对所有软件包信息（如所有 Debian 软件包中的`.desktop` 文件）编制索引。它只要知道标签地址，就能在几毫秒内返回结果。

```
$ axi-cache search forensics graphical
5 results found.
Results 1-5:
100% autopsy - graphical interface to SleuthKit
82% forensics-colorize - show differences between files
    ➥ using color graphics
73% dff - Powerful, efficient and modular digital
    ➥ forensic framework
```

```
53% gpart - Guess PC disk partition table, find lost
    ➥ partitions
46% testdisk - Partition scanner and disk recovery
    ➥ tool, and PhotoRec file recovery tool
More terms: colorize partitions file disklabel autopsy
    ➥ digital differences
More tags: admin::forensics security::forensics
    ➥ role::program admin::recovery
    ➥ interface::commandline admin::boot
    ➥ scope::utility
```

还有一些很少使用的功能。例如，`apt-cache policy` 显示软件包源和独立软件包的优先级。另一个例子是 `apt-cache dumpa vail`，它显示所有软件包可用版本的信息。`apt-cache pkgnames` 命令会返回所有在缓存中出现过软件包的名称列表。

8.2.6　故障排除

在跟 Linux 的安装包打交道的过程中，或多或少总会遇到问题。本节我们将介绍一些基本的故障排除方法，并提供一些工具，让你更容易找到解决方案。

处理升级后的问题

尽管 Kali / Debian 的维护者尽了最大努力，但系统升级并不总是像我们希望的那样顺利。新版本的系统可能与以前版本的软件不兼容（例如，默认配置或者数据格式可能在系统升级时发生了改变），尽管软件包维护人员和 Debian Unstable 用户进行了测试，但还是可能有错误被遗漏了。

利用错误报告来处理问题

有时你可能发现新版本的软件根本无法正常工作。这通常发生在应用程序的使用并不是特别广泛且尚未充分测试就发布的情况下。首先要看看Kali 漏洞跟踪器 [1]和位于`https://bugs.debian.org/package`的Debian漏洞跟踪系统 [2]，是否已经报告了问题。如果没有，你应该自己报告这个漏洞（见 6.3 节的详细说明）。如果已经报告了，错误报告和相关消息通常是相关错误的最好信息来源。在某些情况下，修补程序已经发布，并会在错误报告本身中提供修复补丁；然后你可以在本地重新编译损坏软件包的修复版本（参见 9.1 节）。在其他情况下，用户可能已经找到了解决方法，并在这个报告的答复中分享了他们的

1 http://bugs.kali.org
2 https://bugs.debian.org

见解。这些说明可能会帮助你临时解决这个问题，直到修复补丁发布。在遇到问题时最好的情况是，这个软件包已经被修复，并且可以在错误报告中找到详细信息。

降级到可运行的版本

当问题可以通过回退解决（前一个版本可以正常工作）时，你可以尝试降级软件包。在这种情况下，你将需要一个旧版本的软件副本。如果你可以在 apt 配置数据库中找到这个旧版本，则可以使用简单的一行命令进行降级（参见 8.2.2 节）。但是由于 Kali 是滚动发布的，通常你只能找到每个安装包的单一版本。

你还可以尝试查找 Old.deb 文件，并使用 `dpkg` 手动安装它。old.deb 文件可以在多个地方找到。

- 在 APT 缓存的`/ var / cache / apt / archives /`中。
- 在你常用的 Kali 镜像 pool 目录中（会保留用户删除和过期的软件包三至四天，以避免用户没有最新软件包）。
- 如果受影响软件包是由 Debian 提供的，而不是由 Kali 提供的，在 `http://snapshot.debian.org` 上可以找到 Debian 软件包的所有历史版本。

处理损坏的维护脚本

有时因为软件包的一个维护脚本运行失败（通常 `postinst` 脚本出错）导致升级被迫中断。在这种情况下，你可以尝试通过编辑有问题的脚本来判断问题，并借此解决此问题。

在这里，我们依赖于由维护脚本生成存储在`/ var / lib / dpkg / info /`中的内容，并且我们可以试着去检查和修改它们。

由于维护脚本通常是简单的 shell 脚本，因此可以在可能存在问题的行后面添加一个`-x`行，并使其重新运行（使用 `dpkg --configure -a` 重新运行 `postinst`），看看究竟发生了什么以及哪里失败了。反馈的输出内容也可以很好地补充你在提交错误报告时需要的一些信息。

有了这些新获得的信息，你甚至可以修复潜在的问题，或者将失败的命令转换为一个可用的命令（例如，在行尾添加|| `true`）。

注意，这个方法不适用于因 `preinst` 安装失败导致的问题，因为在安装包之前，该脚本也会执行，而此时程序尚未处于最终状态，所以还是会报出错误。这个方法仅适用于因 `postrm` 和 `prerm` 引起的错误，尽管你可能需要执行一个软件包删除（或者升级）操作来触发它们。

dkpg的日志文件

dpkg 工具在/var/log/dpkg.log 中保存了所有操作的日志。这个日志非常冗长，因为它详细记录了每个软件包的所有阶段。其除了提供跟踪 dpkg 行为的方法之外，还有助于记录系统升级发展历史。你可以找到每个软件包安装或更新时做的任何改变以及确切的时刻，这些信息在理解更新行为的变化时是非常有用的。另外，所有版本都被记录下来了，使用 chagelog.Debian.gz 可以容易地交叉检查出有问题的安装包，甚至可以在线提交错误报告。

```
# tail /var/log/dpkg.log
2016-12-22 09:04:05 status installed kali-linux-gpu:amd64 2016.3.2
2016-12-22 09:20:07 startup packages remove
2016-12-22 09:20:07 status installed kali-linux-gpu:amd64 2016.3.2
2016-12-22 09:20:07 remove kali-linux-gpu:amd64 2016.3.2
2016-12-22 09:20:07 status half-configured kali-linux-gpu:amd64 2016.3.2
2016-12-22 09:20:07 status half-installed kali-linux-gpu:amd64 2016.3.2
2016-12-22 09:20:07 status config-files kali-linux-gpu:amd64 2016.3.2
2016-12-22 09:20:07 status config-files kali-linux-gpu:amd64 2016.3.2
2016-12-22 09:20:07 status config-files kali-linux-gpu:amd64 2016.3.2
2016-12-22 09:20:07 status not-installed kali-linux-gpu:amd64
```

使用apt --reinstall和aptitude重新安装软件包

当你因为删除或修改某些文件而损坏系统时，纠正该错误最简单的方法是重新安装受影响的软件包。不幸的是，系统认为软件包已经安装，而拒绝重新安装它。为了避免这种情况，可以使用 apt 和 apt-get 命令的--reinstall 选项来安装。以下命令即使在 postifx 已经存在的情况下也可以重新安装：

```
# apt --reinstall install postfix
```

aptitude 命令行稍有不同，但使用 aptitude reinstall postfix 命令和以上例子的效果是相同的。dpkg 命令不会阻止重新安装，但很少直接这么用它。

不要使用 apt--reinstall 从一次攻击事件中恢复	在受到攻击之后，使用 apt—reinstall 命令重新安装包，并不能覆盖系统原有的内容。 在受到攻击之后，你不能信任任何东西：dpkg 和 apt 可能都已经被恶意程序所控制，它们将不按照你的意愿重装程序。攻击者也可能在 dpkg 的控制之外改变或者创建文件。

请记住，你也可以使用 apt 来指定特定的发行版，这意味着你可以滚动回到一个比较旧的版本（例如，你知道这个版本运行得很稳定），前提是 source.list 文件的其中一个源仍然可以使用这个版本：

```
# apt install w3af/kali-rolling
```

利用--force- *修复中断的依赖关系

如果你不小心，使用—force- *系列命令和其他一些错误设置可能会导致系统混乱、apt 系列命令拒绝工作等。实际上，其中的一些选项是用来允许在不满足依赖关系或出现冲突时安装程序包的。结果导致从依赖关系的角度来看这个系统是存在问题的，除非使系统恢复到正常状态（通常包括安装缺失的依赖项或删除有问题的程序包），否则 apt 命令将拒绝执行任何操作。在安装新版本的 rdesktop 时忽略它对更新版本 libc6 的依赖，通常会提示这样的信息：

```
# apt full-upgrade
[...]
You might want to run 'apt-get -f install' to correct these.
The following packages have unmet dependencies:
rdesktop: Depends: libc6 (>= 2.5) but 2.3.6.ds1-13etch7 is installed
E: Unmet dependencies. Try using -f.
```

如果你是一位果断的管理员，并且确信你的分析是正确的，则可以选择忽略这个依赖或冲突，并使用相应的--force- *选项来进行操作。在这种情况下，如果你希望能够继续使用 apt 或 aptitude，则必须编辑/ var / lib / dpkg / status，删除或修改依赖关系或冲突。

这个操作是一个有破坏性的操作，除非在极端必要的情况下，原则上不应该这样使用。很多时候，更合适的解决方案，还是重新编译引起问题的软件包，或从提供 backports（backports 是指较新的版本，特别是适合在较旧的环境中重新编译并正常工作）的存储库来安装一个新版本（可能已更正）。

8.2.7 前端：aptitude和synaptic

APT 是一个 C++程序，其代码主要位于 libapt-pkg 共享库中。正是由于 APT 使用了这个共享库，而共享库代码可以很容易地被引用，这才为（前端）用户界面的出现提供了可能。从历史的角度看，apt-get 只是作为 libapt-pkg 的前端而被设计的，但它的成功掩

盖了这个设计初衷。随着命令行界面工具如 apt 和 apt-get 的普及，各种图形界面也被开发出来。下面我们介绍两个图形界面工具：aptitude 和 synaptic

aptitude

如图 8.1 所示，aptitude 是一个交互式程序，可以在命令行控制台窗口上，以半图形模式使用它。使用它可以浏览已安装和可用软件包的列表，查找所有信息，然后选择要安装或删除的软件包。该程序是专门为管理员使用而设计的，所以它的默认操作比 APT 更合理，而且它的界面也更容易理解。

aptitude 会按软件状态（已安装、未安装或已安装但不可用）排序显示软件包列表，我们还可以通过其他视图查看任务、虚拟包以及最近新出现系统上的各个软件包。为了便于浏览查看，还可以使用各种其他视图。

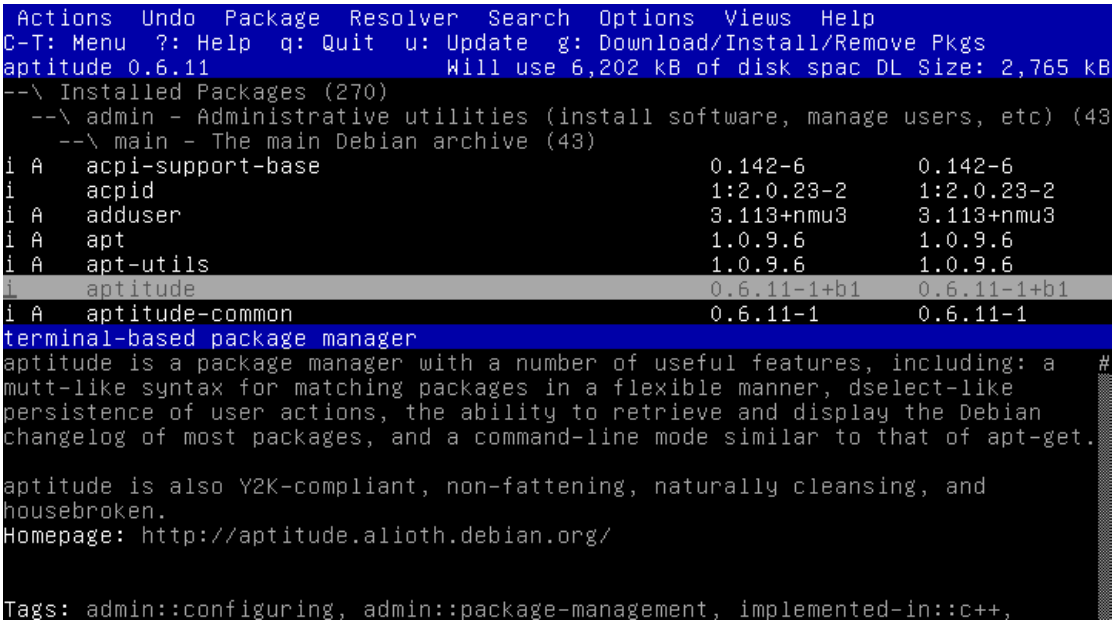


图8.1　aptitude程序包管理器

在正常情况下，aptitude 会在屏幕上显示出一个软件包和包类型的列表。其按照树形结构组织类型，其分支可以用 Enter 键或者“[”和“]”来展开和折叠。+键应该用于标记要安装的软件包，-键将其标记为删除，_键标记为完全清除。注意，这些操作标志的键值也可以用于一个软件类别，在这种情况下，相应的操作会应用于该类别的所有包。u 键对应更新可

用软件包列表，Shift + u 组合键准备全系统升级。g 键切换到请求更改的摘要视图（再次键入 g 表示保存更改并使其生效），然后按 q 键退出当前视图。如果你处于初始视图，按 q 键将关闭软件。

aptitude 文档

本节不会过多介绍如何使用 aptitude，而是着重介绍一些基本使用方法。因为 aptitude 有相当好的文档记录，如果你想了解关于它的更多介绍，建议你阅读 *aptitude-doc-en* 软件包中的完整手册：

➨ file:///usr/share/doc/aptitude/html/en/index.html。

如果需要搜索安装包，可以键入“/”然后跟上搜索内容。这个搜索功能除了可以搜索软件包名之外，还可以搜索软件包描述中的关键字（添加~b 然后添加关键字），也可以去匹配软件包 section 中的内容（使用~s），或者通过添加其他的一些参数，尝试找到匹配关键字的字段。同样我们还可以通过键入 l（limit 的意思），以限制模式在已显示软件包中再一次筛选，以便找出我们想要的。

另外使用 aptitude 工具来管理 Debian 的 automatic flag 安装包（参阅 8.3.4 节）是一件轻松简单的事情。可以通过它浏览已安装软件包的列表，并使用 Shift + m 组合键将软件包标记为自动的，也可以使用 m 键删除标记。自动包在软件包列表中显示为“A”。它还提供了一个直接显示主机上正在运行软件包的视图。在这个视图中你还可以把 l 键（激活过滤器模式）和~i! ～M 联合使用。这组命令将帮你筛选出已安装（~i）但未标记为自动（! ～M）的软件包。

在命令行界面上使用 aptitude

aptitude 的大部分功能都可以通过命令行界面来使用。这些命令行 apt-get 和 apt-cache 的普通用户很熟悉。

其实 aptitude 的高级功能也可以在命令行上实现。一样可以用命令行方式对软件包进行搜索。例如，要把“手动安装”软件包列表中的软件包都变成自动的，并且如果你知道本地已安装的程序不需要什么特定的库或 Perl 模块，就可以使用如下这个命令，将相应的软件包标记为自动的：

```
# aptitude markauto '~slibs|~sperl'
```

在这里，可以清楚地看到 aptitude 搜索模式系统的强大，它可以立即选择 libs 和 perl section 中的所有软件包。

注意！如果某些软件包被标记为自动的，并且没有其他软件包依赖于这些软件包，则这些软件包将会（在确认请求之后）立即被 aptitude 移除。

管理推荐、建议和任务

aptitude 另一个有趣的特点是它会协助用户安装软件包之间的推荐，同时也允许用户根据自己的需求拒绝安装它们。例如，gnome 软件包推荐安装 gdebi（或者其他软件）。当你选择前者进行安装时，后者也将被选中（如果尚未安装在系统上，则将其标记为自动的）。这时如果键入 g 就可以看得很明白。在摘要视图会显示 gdebi 是为了满足依赖关系而需要被自动化安装的待安装项，但是，在确认安装之前，你可以取消选择，不安装它。

注意，推荐跟踪功能不适用于升级过程。例如，如果一个新版本的 gnome 推荐一个以前没有推荐的软件包，这个软件包将不会被标记为安装。但是，它将在升级列表中被列出，以便让管理员可以选择是否安装。

软件包之间的建议安装也考虑了，但也要根据实际的情况来进行判断。例如，因为 gnome 建议安装 dia-gnome，后者将会以待定（由其他软件包建议的包的分类）的操作状态显示在视图上。这样，它很容易被看到，并由管理员决定是否采纳建议。由于这只是一个建议，而不是一个依赖或需求，因此不会被自动安装，而是需要人工干预（因此软件包不会被标记为自动安装）。

同样，aptitude 非常聪明地使用了“任务”这个概念。由于任务可以在软件包列表视图中作为一个类别来操作，所以你可以选择一个完整的任务进行安装或删除，或是浏览任务中包含的软件包列表并选择一个较小的操作子集来执行这个任务。

更好的求解算法

总之，aptitude 在解决困难情况方面有比 APT 更精细的处理办法。当需要进行一系列操作，并且这些操作可能导致一个系统混乱时，aptitude 将评估几种可能的情景，并按照相关性递减的顺序呈现。但是，这些算法并不是万无一失的。幸运的是，我们还可以手动来选择要执行的操作。当选择的操作可能会导致矛盾时，程序会返回显示那些有问题的包（可以通过按 b 键直接导航到这些包并进行操作）。然后可以手动建立一个解决方案。特别是，可以通过使用 Enter 键选择访问包不同的可用版本。如果发现其中一个版本可以解决此问题，应该毫不犹豫地使用该版本。当问题软件包数量减少到零时，可以回到最后一次检查时的待执行操作的界面，然后再应用此配置进行安装。

aptitude 日志　像 dpkg 一样，aptitude 在其日志文件（/var/log/aptitude）中保留一个已执行操作的踪迹。但是，由于这两个命令的工作方式截然不同，因此在其各自日志文件中也很难找到相同的信息。dpkg 只是一步一步地记录在单个软件包上执行的所有操作，而 aptitude 会提供更丰富的高级操作视图，如系统层面的升级操作等。

注意！此日志文件仅包含 aptitude 执行的操作摘要。如果使用其他前端（甚至

dpkg 本身），那么 aptitude 的日志将仅包含操作的部分视图，所以不能依靠它来构建系统历史记录。

synaptic

Synaptic 是一个图形软件包管理器，它基于 GTK+和 GNOME，具有一个干净而高效的图形化界面（如图 8.2 所示）。它的多种即用型过滤器可方便快速地访问新提供的软件包、已安装软件包、可升级软件包、已过时的软件包等。如果浏览找到这些列表，可以直接在列表中选择在软件包上要执行的操作（安装、升级、删除、清除等）。这些操作不是立即执行的，而是放入任务列表中。只需单击一个按钮，然后确认操作，随即所有操作就会一起执行。

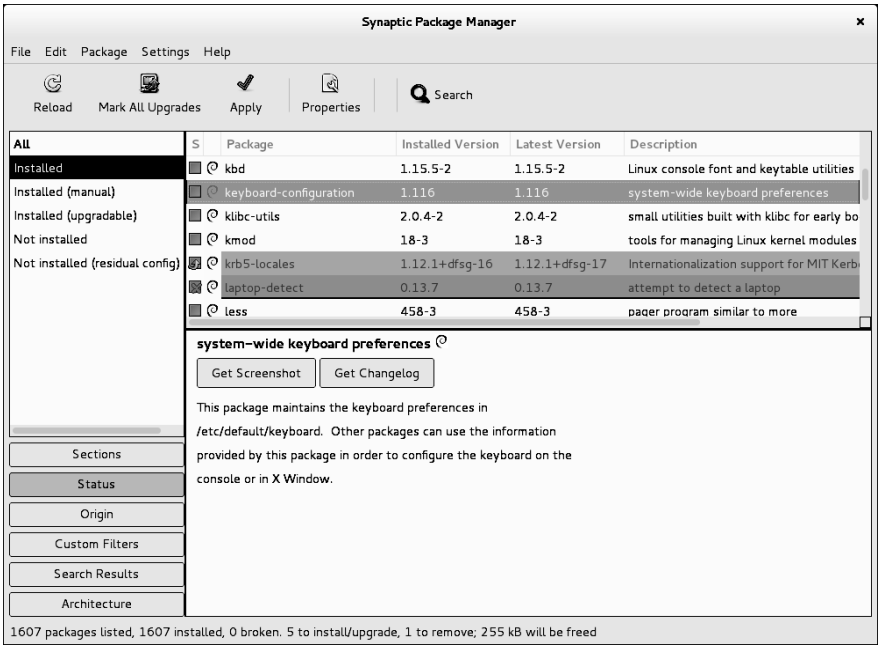


图8.2　synaptic包管理器

8.3　APT高级配置和应用

现在可以深入讨论一些更高级的话题了。首先，我们来看 APT 的高级配置，在这里可以设置很多 APT 工具的功能选项。然后，展示如何控制软件包优先级，以及如何定制个人更新和升级策略。本节还会展示如何在多个软件发行方选择软件来源，以便可以最好地使用来自

不同发行方的软件包。

下面，我们讲解如何跟踪自动安装的软件包，以便可以通过依赖关系，管理各个安装的软件包。还将介绍软件包关于各种体系架构的相关知识，以便去使用各种硬件体系架构的软件包。最后很重要的一点，我们将聊聊加密协议，以及可以保障通信安全的几个工具。

8.3.1　APT配置

在我们深入了解 APT 的配置之前，先花点时间讨论一下 Debian 系统的配置机制。在过去，系统配置是由专用的配置文件处理的。但是随着时代的发展，在像 Debian 和 Kali 这样的现代 Linux 系统中，带有.d 后缀的配置目录变得越来越常用。每个目录代表了被分割成多个文件的配置文件。从这个意义上说，`/etc/apt/apt.conf.d/`中的所有文件都是 APT 的配置指令。APT 按字母顺序加载这些配置文件，以便后面的文件可以修改先前文件中已经定义的配置元素。

这种结构为管理员和软件包维护人员带来了很不错的灵活性，允许他们直接通过添加文件来进行软件配置更改，而无须修改现有配置文件。这对于软件包维护者特别有用，因为他们可以使用这种方法，来调整其他软件的配置文件，从而使这个软件可以和其他软件能够完美共存，而不会破坏 Debian 安全策略中，明确禁止一个软件修改其他软件包配置文件这条规定。由于`.d` 的配置机制，所以不必手动执行软件包`/ usr / share / doc / package / README.Debian` 文件中的多个软件包配置指令，因为直接在配置文件中写好安装指令就行了。

注意从.d 目录生成的配置文件

APT 自己的配置文件就是`/etc/apt/apt.conf.d` 目录，但并不是所有程序的配置文件都是这样。某些应用程序（例如 `exim`）的`.d` 目录是由 Debian 特定添加的，用作应用程序动态生成标准配置文件的输入文件。在这种情况下，软件包会提供一个 `update- *`命令（例如，`update-exim4.conf`），它将连接`.d` 目录中的文件并重写主配置文件。

在这种情况下，你不能手动直接编辑主配置文件，因为你更改的那部分在下次执行 `update- *`命令时将会丢失，并且在编辑了对应的`.d` 文件之后，别忘了运行一下之前重写主配置文件的命令（否则你的更改将不会生效）。

了解了`.d` 配置机制之后，我们来谈谈如何利用它来配置 APT。正如我们已经讨论过的，你可以通过命令行参数来改变 APT 的行为，比如下面的例子，可以执行 `zsh` 的强制覆盖安装：

```
# apt -o Dpkg::Options::="--force-overwrite" install zsh
```

显然这是个笨办法，尤其是如果你经常使用这样的命令，这实在是太麻烦了。你可以在/etc/apt/apt.conf.d/目录下添加包含这一指令的文件，通过调整.d 目录配置结构的方式来配置 APT 的某些操作。例如，可以把这段命令写到/etc/apt/apt.conf.d/目录下的一个文件中，生成一个叫 99local 的文件，这个文件的名字有点随意，但它只是一个通用的名称，使用 local 或者 99local 其实都无所谓：

```
$ cat /etc/apt/apt.conf.d/99local
Dpkg::Options {
   "--force-overwrite";
}
```

还有许多其他有用的配置选项，我们不能把所有这些配置选项都介绍一遍，但需要讲一下涉及网络连接的设置方式。例如，如果你希望只通过代理访问 Web，可以在配置文件中添加一行 Acquire:: HTTP::proxy "http://yourproxy:3128"。对于 FTP 代理，可以使用 Acpuire :: ftp :: proxy"ftp://yourproxy"。

要了解更多配置选项，可以使用 man apt.conf 命令（有关手册页的详细信息，参见 6.1.1 节的介绍）查看 apt.conf（5）手册页。

8.3.2 包优先级管理

管理各个软件源的优先级是 APT 配置中非常重要的一部分。例如，你可能需要在 Kali 系统中安装几个来自 Debian Unstable 或 Debian Experimental 源更新更好用的软件包。你可能需要为每个可用软件包分配一个优先级（同一个软件的包根据它的版本和提供软件的来源可以有几个优先级）。这些优先级将影响 APT 的行为。对于每个软件，它将始终选择具有最高优先级的版本（除非该版本比已安装的当前版本旧，且其优先级小于 1000）来进行安装。

APT 定义了几个默认优先级。每个已安装软件包版本的优先级为 100。未安装版本的默认优先级为 500，但如果软件的源和已安装的源相同（通过-t 选项或 apt::Default-Release 设置），则它的优先级会提升到 990。

可以通过在/ etc / apt / preferences 文件中以软件包名称、版本、源等元素为影响因子来调整新的软件包优先级计算标准。

在软件包安装的过程中，APT 更优先选择新版的软件包，除非旧软件包的优先级高于 1000，否则 APT 将永远不会安装较旧版本的软件包（即版本号低于当前安装软件包的软件

包）。

同时 APT 总是会选择优先级最高的软件包进行安装，如果两个软件包的优先级一样，那么 APT 会选择版本更新的那个（其版本号最高）。如果连版本号也一样，但内容不同，APT 会选择安装没被安装的那个（该规则适用于覆盖软件更新后但没有增加修订版本号的情况，这一点通常是必须了解的）。

如果包的优先级低于 0，则它永远不会被安装。假如一个包的优先级在 0～100 之间，只有在没有安装其他版本的情况下，它才会被安装。假如安装包优先级在 100～500 之间，如果没有安装更新的版本，或者在其他发行版中没有可用的版本，其才会被安装。假如一个包的优先级在 501～990 之间，而且没有安装更新的版本，或在目标发行版中没有可用版本，其才会被安装。对于优先级在 990～1000 之间的，除非已安装版本号比此版本的版本号更高，否则将安装此软件包。优先级大于 1000 的安装包总是最优先被安装。

当 APT 查看`/etc/apt/preferences`时，它会首先考虑特殊条目（通常是指定条件的相关包），然后才是普通软件的条目（例如某软件发行源的所有包）。如果存在多个条目，则使用最先匹配的。判断的关键标准是包的名字和对应的源。每个软件包源都由在 APT 下载软件安装包文件时一起下载的 `Release` 文件进行标识。这些文件指出了软件的来源，来自 Kali 官方镜像的软件包的来源标志通常是“Kali”，Debian 官方镜像中的软件包的标志是“Debian”，软件源也可以是第三方存储库的个人或机构名称。`Realease` 文件还提供软件分发源的名称及其版本。下面我们通过一些实际案例，来理解它的语义结构。

Kali-Bleeding-Edge 和 Debian Experimental 的优先级

即使 sources.list 文件中列出了 `kali-bleeding-edge` 或 `Debian experimental` 这两个源，但它们的相应软件包还是几乎不会被安装，因为它们的默认 APT 优先级为 1。这当然是一个特殊情况，旨在防止用户误装了 `bleeding edge` 的软件包。如果你知道安装 `bleeding edge` 源软件包时存在的风险和可能出现的问题，你仍然想要去安装，只有通过输入 `apt install package / kali-bleeding-edge` 这样的命令，才能从这个源安装软件包。或者你可以（不推荐）像对待其他软件源一样，也给 `kali-bleeding-edge / experimental` 的包设置一个 500 的优先级。这可以通过在`/ etc / apt / preferences` 中添加一个特定条目来完成：

```
Package: *
Pin: release a=kali-bleeding-edge
Pin-Priority: 500
```

假设你只想使用来自 Kali 的软件包，只有在明确请求时才安装 Debian 软件包。可以在`/`

etc / apt / preferences 文件（或/etc/apt/preferences.d/中的任何文件）中写入以下条目：

```
Package: *
Pin: release o=Kali
Pin-Priority: 900

Package: *
Pin: release o=Debian
Pin-Priority: -10
```

在最后的两个例子中，我们看到了 a = kali-bleeding-edge，它定义了所选软件源的名称；o = kali 和 o = debian，它们将范围限制为来源于 Kali 和 Debian 的包。

现在我们假设你有一台服务器，其中几个服务都依赖于 Perl 5.22 这个版本，并且你想确保升级不会使 Perl 升级到另一个版本。可以使用这个条目：

```
Package: perl
Pin: version 5.22*
Pin-Priority: 1001
```

这个配置文件的参考文档可以在使用 man apt_preferences 命令显示的 apt_preferences（5）手册页中找到。

在/ etc / apt / preferences 中添加注释

/ etc / apt / preferences 中没有提供注释的正式语法，但是可以通过在每个条目中预先添加一个或多个解释字段来提供一些文本描述：

```
Explanation: The package xserver-xorg-video-intel
provided
Explanation: in experimental can be used safely
Package: xserver-xorg-video-intel
Pin: release a=experimental
Pin-Priority: 500
```

8.3.3 同时运行几个软件分发源

APT 是一个非常棒的工具，它还能让我们安装其他发行版的软件。例如，在安装 Kali Rolling 的系统时，你可能需要尝试使用 Kali Dev、Debian Unstable 或 Debian Experimental 中提供的软件包，但又不想与系统初始状态偏离太远。

尽管安装来自不同发行源的软件可能会引发问题，但 APT 仍然可以很好地管理这种共存

状态，并且能非常有效地限制风险（假设软件包的依赖关系是准确的）。首先，要在`/etc/apt/sources.list` 中配置所有发行版的源，并使用 `apt::Default-Release` 来定义你更偏好的软件分发源（参见 8.2.3 节）。

假设 Kali Rolling 是你的默认分发源，但是 Kali Dev 和 Debian Unstable 也列在 sources.list 文件中。在这种情况下，可以使用 `apt install package / unstable` 从 Debian Unstable 安装软件。如果由于某些依赖关系没有满足而导致安装失败，可以让 APT 通过`-t unstable` 来安装属于 Unstable 的依赖项。

在这种情况下，如果要升级（`upgrade` 和 `full-upgrade`），只会从 Kali Rolling 这个分发源来进行更新升级，除了从其他分发源安装的软件，它们会从对应的分发源中更新软件。我们可以根据下面 APT 设置默认优先级的原则来解释这种行为。大家不用担心，可以大胆地使用 `apt-cache policy` 命令（参见下面的“使用 `apt-cache policy`”的介绍）来验证给定的优先级。

一切都依赖于，APT 只考虑版本号高于或等于已安装包的软件包（假设并未使用`/ etc / apt / preferences` 将某些软件包的优先级强制设置为高于 1000）。

使用 `apt-cache policy`　利用 `apt-cache policy` 可以查看每个源的优先级。也可以使用 `apt-cache policy package` 查看具体某软件所有可用版本的优先级。

假设已经安装了 Kali Rolling 发行版中版本 1 的软件包，并且该软件的版本 2 和 3 分别在 Kali Dev 和 Debian Unstable 中可用。已安装版本的优先级为 100，但 Kali Rolling（同发行源）中的版本的优先级为 990（因为它是目标源内的软件包）。Kali Dev 和 Debian Unstable 中的软件包的优先级为 500（未安装版本的默认优先级）。因此，获胜者是版本 1，优先级为 990。

升级包选择还是在 Kali Rolling 源。

我们再来看一个例子。如果当前状态是已经安装了来自 Kali Dev 发行版该软件的版本 2，版本 1 在 Kali Rolling 和 Debian Unstable 的版本 3 中可用。版本 1（优先级为 990，因此低于 1000）的优先级因为低于安装版本而被丢弃。只剩下版本 2 和 3 了，优先级都是 500。面对这种情况，APT 选择了最新的版本，Debian Unstable。如果不希望将从 Kali Dev 安装的软件包迁移到 Debian Unstable，则必须为来自 DebianUnstable 的软件包指定一个低于 500（例如 490）的优先级。可以通过修改`/ etc / apt / preferences` 实现：

```
Package: *
Pin: release a=unstable
Pin-Priority: 490
```

8.3.4 跟踪自动安装的包

跟踪通过依赖关系自动安装的包是 APT 的一个基本功能，这些包被称为“自动包”，它们通常是各种库。

有了这些信息，当软件包被移除时，软件包管理器可以获得一个不再需要的自动包列表（因为没有其他哪个安装的软件包依赖它们了）。`apt autoremove` 命令可以用来清除这些自动包。`aptitude` 没有这个选项，因为它会自动清除没用的依赖。在所有情况下，这两个软件都会提示用户这些操作的具体细节并列出将会受到影响的软件包。

将自己不直接使用的软件包标记为自动包是一种很好的习惯，以便在不再需要时能够将其卸载。`apt-mark auto package` 会将包标记为自动包，而 `apt-mark manual package` 则相反。`aptitude markauto` 和 `aptitude unmarkauto` 具有同样的功能，但是这两个命令可以一次性标记多个包（参见 8.2.7 节）。

`aptitude` 基于控制台的交互界面也使得查看许多包中的自动标记变得很容易。

你可能想知道一个软件包是由于什么原因被安装在了系统上。可以使用 `aptitude why package`（`apt` 和 `apt-get` 没有类似的功能）命令行查看此信息：

```
$ aptitude why python-debian
i aptitude Recommends apt-xapian-index
i A apt-xapian-index Depends python-debian (>= 0.1.15)
```

8.3.5 Multi-Arch支持

所有 Debian 软件包的控制信息都有一个 Architecture 字段。该字段可以包含“all”（说明此软件适合所有架构），或者字段内容标明此软件包适用体系架构的名称（如 amd64 或 armhf）。在后一种情况下，如果由 `dpkg` 显示的此软件的体系架构与由 `dpkg --print-architecture` 返回的主机体系架构相匹配，`dpkg` 则安装它。

这个限制可以确保你不会因为选择了不正确体系架构的软件包而导致编译无法通过。除了（某些）计算机可以（通过本机的特殊设计，如使 amd64 系统可以运行 i386 二进制文件，或通过仿真器）运行多种体系架构的二进制文件之外，其他情况都需要正确匹配相应的体系架构。

启用multi-arch 多体系架构支持

有multi-arch作支持，`dpkg`可以接受用户安装在当前系统上其他系统体系架构的软件包。

使用pkg --add-architecture 命令可以方便地完成设置，就像下面的例子一样，需要将i386 架构添加到amd64 系统中，以便可以使用Wine [1]运行Windows应用程序。相应的可以使用dpkg --remove-architecture命令来取消对该外部架构的支持，但是需要在没有此架构软件包继续保持安装的情况下才能进行此操作。

```
# dpkg --print-architecture
amd64
# wine
it looks like wine32 is missing, you should install it.
multiarch needs to be enabled first. as root, please
execute "dpkg --add-architecture i386 & apt-get update &
apt-get install wine32"
Usage: wine PROGRAM [ARGUMENTS...] Run the specified program
       wine --help Display this help and exit
       wine --version Output version information and exit
# dpkg --add-architecture i386
# dpkg --print-foreign-architectures
i386
# apt update
[...]
# apt install wine32
[...]
Setting up libwine:i386 (1.8.6-5) ...
Setting up vdpau-driver-all:i386 (1.1.1-6) ...
Setting up wine32:i386 (1.8.6-5) ...
Setting up libasound2-plugins:i386 (1.1.1-1) ...
Processing triggers for libc-bin (2.24-9)
# wine
Usage: wine PROGRAM [ARGUMENTS...] Run the specified program
       wine --help Display this help and exit
       wine --version Output version information and exit
# dpkg --remove-architecture i386
dpkg: error: cannot remove architecture 'i386' currently in use by the database
# dpkg --print-foreign-architectures
i386
```

如果 dpkg 配置可以支持其他体系架构的软件包，运行 APT 后会自动检测，并会在更新过程中下载相应的包文件。

其他体系架构的安装包可以使用 apt install package:architecture 命令进行安装。

1 https://www.winehq.org/

在 amd64 上使用专有的 i386 二进制文件	multi-arch 多体系架构在很多场景中都可以应用，但最常见到的一个场景是在 64 位系统（amd64）上执行 32 位二进制文件（i386），特别是，有几个很流行的专有应用程序（如 Skype）只提供了 32 位版本的情况。

multi-arch 带来的变化

为了使 multi-arch 多体系架构实际有用并且更易用，所有的库必须重新打包，并将它们复制到特定架构的目录下，以便不同体系架构的软件可以并行安装。如果更新的包中如果包含 `Multi-Arch:same` 头字段，则表示该包的各种架构版本都可以一起安装。（并且这些软件包只能满足相同体系架构的软件包的依赖关系。）

```
$ dpkg -s libwine
dpkg-query: error: --status needs a valid package name but 'libwine' is not:
    ➥ ambiguous package name 'libwine' with more than one installed instance

Use --help for help about querying packages.
$ dpkg -s libwine:amd64 libwine:i386 | grep ^Multi
Multi-Arch: same
Multi-Arch: same
$ dpkg -L libgcc1:amd64 |grep .so
[...]
/usr/lib/x86_64-linux-gnu/wine/libwine.so.1
$ dpkg -S /usr/share/doc/libwine/copyright
libwine:amd64, libwine:i386: /usr/share/doc/libwine/copyright
```

有一点需要注意，`Multi-Arch:same` 软件包需具备能够明确识别架构的名称。同一软件包的不同架构之间可以共享文件；当文件被共享的时候，`dpkg` 能够确保所有包的文件正常。另外，这类软件包的不同架构必须保持相同版本，因此它们必须一起升级。

随着 multi-arch 的使用，在处理依赖方面也随之出现了一些有意思的问题。为满足依赖关系，必须使用标记有`"Multi-Arch:foreign"`的软件包，或该包的架构与依赖需求的体系架构完全一样。可以使用 `package:any` 语句减弱依赖关系，以使任何体系架构都可以被接受，但 foreign 软件包仅能满足有“`Multi-Arch: allowed`”标识的依赖。

8.3.6 检查包的真实性

系统升级是非常敏感的操作，你肯定希望确保只安装来自 Kali 官方源的更新。如果你正在使用的 Kali 更新源已被攻陷，电脑黑客可以将恶意代码添加到某个合法软件包内。如果系

统安装了这样的软件包，黑客就可以对你的电脑进行任何操作，包括泄露密码或机密信息。为了规避这种风险，Kali 提供了一个防篡改印章机制，以保证在安装的时候，软件包是官方开发的版本，并且没有被第三方修改过。

这个印章使用哈希加密和签名。被签名的文件叫 Release，它由 Kali 镜像服务器提供。该文件包含了 Packages 文件列表（包括它们的压缩表单、Packages.gz 和 Packages.xz 以及增量版本）和它们的 MD5、SHA1、SHA256 哈希值，以确保这些文件没有被篡改过。这些 Packages 文件包含了镜像中可用的 Debian 软件包列表，以及它们的哈希值，从而确保软件包本身的内容也不会被篡改。

可使用 `apt-key` 工具来管理这些受信任的 keys。它维护一个 GnuPG 公开密钥的 keyring 文件，该文件被用来校验 `Release.gpg` 中的签名。它允许手动添加 key（当需要非官方的镜像时），但通常只需使用 Kali 官方 key。这些 key 由 debian-archive-keyring 这个包进行自动更新（将相应的密钥放在`/etc/apt/trusted.gpg.d` 中）。但是，首次安装这一特定软件包时需要谨慎，即使该软件包像任何其他软件包一样被签名，但该签名不能在外部进行验证。

因此，谨慎的管理员会先检查导入密钥的指纹，然后再判断是否相信它们，进而安装新的软件包：

```
# apt-key fingerprint
/etc/apt/trusted.gpg.d/debian-archive-jessie-automatic.gpg
------------------------------------------------------------
pub 4096R/2B90D010 2014-11-21 [expires: 2022-11-19]
    Key fingerprint = 126C 0D24 BD8A 2942 CC7D F8AC 7638 D044 2B90 D010
uid                 Debian Archive Automatic Signing Key
                    (8/jessie)<ftpmaster@debian.org>

/etc/apt/trusted.gpg.d/debian-archive-jessie-security-automatic.gpg
--------------------------------------------------------------------
pub 4096R/C857C906 2014-11-21 [expires: 2022-11-19]
    Key fingerprint = D211 6914 1CEC D440 F2EB 8DDA 9D6D 8F6B C857 C906
uid                 Debian Security Archive Automatic Signing Key
                    (8/jessie)<ftpmaster@debian.org>

/etc/apt/trusted.gpg.d/debian-archive-jessie-stable.gpg
-------------------------------------------------------
pub 4096R/518E17E1 2013-08-17 [expires: 2021-08-15]
    Key fingerprint = 75DD C3C4 A499 F1A1 8CB5 F3C8 CBF8 D6FD 518E 17E1
uid                 Jessie Stable Release Key<debian-release@lists.debian.org>

/etc/apt/trusted.gpg.d/debian-archive-squeeze-automatic.gpg
```

```
------------------------------------------------------------
pub 4096R/473041FA 2010-08-27 [expires: 2018-03-05]
    Key fingerprint = 9FED 2BCB DCD2 9CDF 7626 78CB AED4 B06F 4730 41FA
uid                 Debian Archive Automatic Signing Key
                    (6.0/squeeze)<ftpmaster@debian.org>

/etc/apt/trusted.gpg.d/debian-archive-squeeze-stable.gpg
------------------------------------------------------------
pub 4096R/B98321F9 2010-08-07 [expires: 2017-08-05]
    Key fingerprint = 0E4E DE2C 7F3E 1FC0 D033 800E 6448 1591 B983 21F9
uid                 Squeeze Stable Release Key<debian-release@lists.debian.org>

/etc/apt/trusted.gpg.d/debian-archive-wheezy-automatic.gpg
------------------------------------------------------------
pub 4096R/46925553 2012-04-27 [expires: 2020-04-25]
    Key fingerprint = A1BD 8E9D 78F7 FE5C 3E65 D8AF 8B48 AD62 4692 5553
uid                 Debian Archive Automatic Signing Key
                    (7.0/wheezy)<ftpmaster@debian.org>

/etc/apt/trusted.gpg.d/debian-archive-wheezy-stable.gpg
------------------------------------------------------------
pub 4096R/65FFB764 2012-05-08 [expires: 2019-05-07]
    Key fingerprint = ED6D 6527 1AAC F0FF 15D1 2303 6FB2 A1C2 65FF B764
uid                  Wheezy Stable Release Key<debian-release@lists.debian.org>

/etc/apt/trusted.gpg.d/kali-archive-keyring.gpg
------------------------------------------------
pub 4096R/7D8D0BF6 2012-03-05 [expires: 2018-02-02]
    Key fingerprint = 44C6 513A 8E4F B3D3 0875 F758 ED44 4FF0 7D8D 0BF6
uid                  Kali Linux Repository<devel@kali.org>
sub 4096R/FC0D0DCB 2012-03-05 [expires: 2018-02-02]
```

当第三方软件源被添加到 `sources.list` 文件时，需要告知 APT 去验证相应的 GPG 认证密钥（否则它会一直提示无法确定来自该软件源的软件包的真实性）。第一步当然是获得公钥。通常情况下，密钥将作为一个小文本文件被下载，就好像我们在下面例子中称为 `key.asc` 的文件。

将密钥添加到可信密钥环中，管理员可以运行 `apt-key add <key.asc` 来手动操作。另一种方法是用 `synaptic` 图形界面。Settings→Preositories 菜单中的 Authentication 选项卡提供了从 `key.asc` 文件导入密钥的功能。

对于更乐于使用专用程序以及想知道可信密钥更多信息的人来说，可以使用 `gui-apt-`

key（软件包和它的命令是一样的名字）这样一个小型图形界面来管理可信密钥环。

一旦相关的密钥放在了密钥环中，APT 将在执行任何有风险的操作之前检查签名，如果不能确定软件包的真实性，则会在前端弹出警告提示。

8.4　软件包参考：深入挖掘Debian软件包系统

现在我们可以深入探索 Debian 和 Kali 软件包系统了。在这里，我们将超越工具和语法，更多地关注软件包系统的具体操作细节。了解这些后台的原理，将帮助你进一步了解 APT 是如何在系统上工作的，让你能够知道如何更加流畅地使用甚至定制自己的 Kali 系统。你可能不会记住本节中的所有内容，但是随着你对 Kali Linux 系统认识的加深，参考资料将会是你最好的帮手。

到目前为止，我们已经了解了如何通过各种工具与 APT 设计接口来进行软件包数据的交互。接下来，我们将深入研究这些软件包，并研究软件包管理工具使用的内部元信息（或其他信息）。

一个简单包的.deb 文件是根据 ar 存档格式，由文本文件、元信息，以及这个归档文件包含的软件包中所有要解压的文件组成的。

```
$ ar t /var/cache/apt/archives/apt_1.4~beta1_amd64.deb
debian-binary
control.tar.gz
data.tar.xz
```

debian-binary 文件是一个文本文件，简单写明了该.deb 文件使用的版本。

```
$ ar p /var/cache/apt/archives/apt_1.4~beta1_amd64.deb debian-binary
2.0
```

control.tar.gz 是归档文件包含的所有可用的元信息。

```
$ ar p /var/cache/apt/archives/apt_1.4~beta1_amd64.deb control.tar.gz | tar -tzf -
./
./conffiles
./control
./md5sums
./postinst
./postrm
./preinst
./prerm
```

```
./shlibs
./triggers
```

最后，data.tar.xz 压缩文件（压缩格式可能不同）包含了软件包安装所需要的文件。

```
$ ar p /var/cache/apt/archives/apt_1.4~beta1_amd64.deb data.tar.xz | tar -tJf -
./
./etc/
./etc/apt/
./etc/apt/apt.conf.d/
./etc/apt/apt.conf.d/01autoremove
./etc/apt/preferences.d/
./etc/apt/sources.list.d/
./etc/apt/trusted.gpg.d/
./etc/cron.daily/
./etc/cron.daily/apt-compat
./etc/kernel/
./etc/kernel/postinst.d/
./etc/kernel/postinst.d/apt-auto-removal
./etc/logrotate.d/
./etc/logrotate.d/apt
./lib/
./lib/systemd/
[...]
```

注意，本例中的 APT 存档缓存中的.deb 软件包，可能与你的存档文件中的文件版本号不同。

下面我们介绍软件包中的元信息，并讲述如何使用它。

8.4.1 control文件

我们先来看看 control.tar.gz 软件包中包含的 control 文件。control 文件包含着有关软件包的最重要信息。该文件使用一个类似电子邮件头的结构，我们可以使用 dpkg -I 命令来查看该文件。例如，APT 的 control 文件信息如下所示：

```
$ dpkg -I apt_1.4~beta1_amd64.deb control
Package: apt
Version: 1.4~beta1
Architecture: amd64
Maintainer: apt Development Team<deity@lists.debian.org>
Installed-Size: 3478
```

```
Depends: adduser, gpgv | gpgv2 | gpgv1, debian-archive-keyring, init-system-
    ➥ helpers (>= 1.18~), libapt-pkg5.0 (>= 1.3~rc2), libc6 (>= 2.15),
    ➥ libgcc1 (>= 1:3.0), libstdc++6 (>= 5.2)
Recommends: gnupg | gnupg2 | gnupg1
Suggests: apt-doc, aptitude | synaptic | wajig, dpkg-dev (>= 1.17.2),
    ➥ powermgmt-base, python-apt
Breaks: apt-utils (<< 1.3~exp2~)
Replaces: apt-utils (<< 1.3~exp2~)
Section: admin
Priority: important
Description: commandline package manager
 This package provides commandline tools for searching and
 managing as well as querying information about packages
 as a low-level access to all features of the libapt-pkg library.
 .
 These include:
 * apt-get for retrieval of packages and information about them
   from authenticated sources and for installation, upgrade and
   removal of packages together with their dependencies
 * apt-cache for querying available information about installed
   as well as installable packages
 * apt-cdrom to use removable media as a source for packages
 * apt-config as an interface to the configuration settings
 * apt-key as an interface to manage authentication keys
```

接下来，我们来了解整个 `control` 文件的内容，以及其各个字段的含义。其中每一个字段都将帮你更好地理解软件包系统，为你提供更加精细的配置控制，并能在你遇到软件包安装和使用问题时提供帮助。

依赖关系：Depends字段

软件包依赖关系被定义在包头中的 Depends 字段中。它包含了软件包正常运行所需要的所有条件，利用这些信息，像 APT 这样的工具就可以安装合适版本的依赖库，对于每个依赖项，都需要限定它的版本。举个例子来说，你需要的软件包 libc6 版本等于或大于“2.15”（写为“`libc6（> = 2.15）`”）。比较版本的运算符如下：

- <<：小于
- <=：小于或等于
- =：等于（注意“`2.6.1`”不等于“`2.6.1-1`”）
- > =：大于或等于
- >>：大于

条件之间以逗号分隔，作用相当于逻辑“and”。竖线（“|”）表示逻辑“or”。“|”的优先级高于“,”。因此，依赖关系“（A or B）and C”被写为 A | B, C。相反，表达式“A or（B and C）”应写为“（A or B）and（A or C）”，因为 Depends 字段不允许用括号来改变逻辑运算符 or 和 and 之间的优先级，因此会写成 A | B,A | C。有关更多信息，请参阅 http://www.debian.org/doc/debian-policy/Ch- relationships.html 上的说明。

依赖关系是保证程序运行的机制，但是 meta-packages 包在这方面有另一个用途。这是一种空的软件包，只含有一些依赖信息，以便于根据 meta-packages 维护者预先选择的一组搭配来更好地完成程序安装过程。就好像，apt install meta-package 会根据 meta-package 的依赖关系自动安装这些有关程序。gnome、kde-full 和 kali-linux-full 等就是使用 meta–packages 来安装依赖关系项的。

Pre–Depends，一个更严格的依赖关系

Pre-Depends 依赖信息放在软件包头部中的“Pre-Depends” 字段中，通常是依赖关系的补充，它的语法跟 Depends 字段一致。通常的依赖表明依赖库的解包和配置都要在待安装的软件配置之前，而 Pre-Depends 规定，在安装这个软件包之前，就必须已经安装好了预依赖程序，并根据预安装配置脚本配置妥当。

Pre-Depends 对于 APT 来说非常苛刻，因为它增加了安装顺序上的约束，因此，如果没有必要就不要使用。甚至，Debian 建议在使用 Pre-Depends 之前，要咨询 debian-devel@lists.debian.org 上面的其他开发者。通常可以找到其他替代解决方案。

Recommends、Suggests 和Enhances 字段

推荐（Recommends）和建议（Suggests）中描述的依赖都不是强制性的。Recommends 中的依赖最重要的作用是大大增加了软件包所提供的功能，但它们对软件运行并不是不可或缺的。Suggests 是更次一级的依赖关系，通常对软件提供一些补充，不安装也完全没问题。

它俩的关系就好像，在安装依赖的时候，除非你确切地知道你为什么不需要这个 Recommends 项，否则最好还是应该安装它们。而 Suggests 却相反，除非你知道为什么需要它们，否则就没有必要安装。

Enhances 字段中的依赖也属于建议性的，但却有不同的含义。它确实是一种建议的安装包，而不是从建议中受益的软件包。它的作用在于，告诉你这个软件可能对其他某个软件有增强的作用。因此，很多程序的各种插件和其他扩展程序都可能出现在与软件相关的建议列表中。虽然 Enhances 字段已经出现好几年了，但是作为最后一个字段在多数时候仍然被 APT

或者 synaptic 等所忽视。其最初的目标是为像 xul-ext-adblock-plus（一个 Firefox 扩展）这样的包提供声明，并且使其出现在 firefox 和 firefox-esr 相关建议包的列表中。

冲突：Conflicts字段

Conflicts 字段表示一个软件包不能与另一个软件包同时安装。最常见的原因是这两个软件包都包含一个相同名称的文件，在相同的传输控制协议（TCP）端口上提供相同的服务，或者会阻碍彼此的操作。

如果一个软件包触发与已安装软件包的冲突，dpkg 将拒绝安装软件包，除非指定新软件包替换已安装软件包，在这种情况下，dpkg 将选择使用新软件包替换旧软件包。APT 就会智能很多。如果你选择安装一个新软件包，但是其触发了冲突，它会自动提供卸载引起问题的软件包的操作。

不兼容性：Breaks字段

Breaks 字段与 Conflicts 字段的效果类似，但是在含义上有些许不同。它表明安装一个软件包会破坏另一个软件包（或其特定版本）。一般来说，这两个软件包之间的不兼容性是短暂的，并且 Breaks 通常明确指明不兼容的软件包版本。

当一个软件包会破坏已安装的软件包时，dpkg 将拒绝安装它，APT 会尝试更新被破坏软件包来解决这个问题（假定这个软件包已经被修复并且因此再次兼容）。

通常，这种情况会出现在系统升级了一个不向后兼容软件包的情况。例如新版本不再适用于旧版本，并且在没有特别规定的情况下导致另一程序出现故障，则是这种情况。Breaks 字段有助于防止这些类型的问题发生。

返回条目：Provides 字段

这个字段涉及一个非常有趣的概念：虚拟软件包。它有很多作用，但是其中两个是特别重要的。第一个是使用一个虚拟包将通用服务与它关联（该包提供服务）。第二个作用是使用一个包完全替换另一个包，并且为了这个目的，它可以满足各种依赖关系。因此可以创建替换包，而不必使用相同的包名。

元包和虚拟包

清楚区分元软件包和虚拟软件包是至关重要的。前者是真正的软件包（包括真正的.deb 文件），其唯一目的是表达依赖关系。

然而，虚拟软件包在物理上并不存在，它们只是根据通用的逻辑标准（例如提供的服务，或与标准程序或预先存在软件包的兼容性），来识别真实软件包的手段。

提供服务

我们来举一个例子，针对第一种情况。在 Linux 中，有好几个包都可以提供 Email 投递服务，比如 `postfix`、`sendmail` 等。这些包虽然名字都不一样，但都会提供同样的服务：`mail-transport-agent`，这个服务就是一个虚拟包。因此，任何需要这种服务的软件包（例如，邮件列表管理器，如 smartlist 或 sympa）只需要在它的依赖中声明它需要 mail-transport-agent，而不用指定一个复杂而且还不完整的解决方案。此外，在同一台计算机上安装两个邮件服务器是不行的，这就是为什么每个程序包都声明与 mail-transport-agent 虚拟包冲突的原因。系统会忽略软件包和它自身之间的冲突，但是这种技术可以有效防止在一个服务器上同时安装两个邮件服务器。

与另一个软件包的互换性

当一个包的内容包含在一个更大的包中时，Provides 字段会变得很有趣。例如，libdigest-md5-perl 在 Perl 5.6 之前是一个可选的软件包，但是在 Perl 5.8 之后就是一个系统默认自带的软件包。在这种情况下，Perl 5.8 版本的运行环境可以声明一个 libdigest-md5-perl 的虚拟包，这样的话，基于 libdigest-md5-perl 这个包旧版本的应用程序仍然可以运行。另外用户升级的时候，libdigest-md5-perl 这个实际包也会被移除，因此系统中老版本的 Perl 已经移除了新版本中自带的这个包中的功能，因此它也就没有存在的必要了，会被自动删除。如图 8.3 所示。

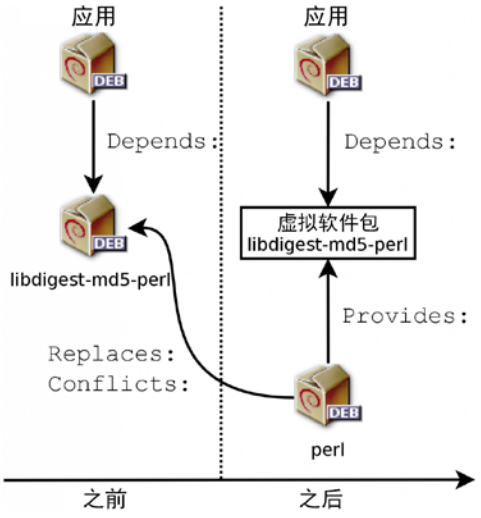


图8.3　利用Provides 字段避免破坏依赖关系

这个功能是非常有用的，因为我们不可能预知技术的发展走向，但我们必须能够适应旧软件重命名和其他自动替换的情况。

替换文件：Replaces字段

Replaces 字段中的信息说明了这个包中的某些文件会替换系统中其他包所属的已存在文件，但是这个替换行为是合法的。如果没有这个字段，dpkg 的安装就会失败，因为可能这个包会破坏当前的系统（虽然你可以使用–force-overwrite 选项来强制覆盖，但是这终究不是一个靠谱的做法）。利用这个字段可以帮我们确定潜在的问题，维护人员可以在安装之前研究并对是否添加这个字段作出自己的判断。

当包名称发生改变，或者这个包已经被包含在另一个包中时，可以使用这个字段来说明。或者当维护者决定由相同源代码包生成多个二进制包，并把它们分配到不同文件时，也会发生这种情况：被替换的文件不再属于旧包，而属于新包。

如果一个包的所有文件都被替换了，则该包被认为是可移除的。最终会使 dpkg 移除完全被替换的包。

8.4.2　配置脚本

除了 control 文件之外，每个 Debian 软件包的 control.tar.gz 压缩文件还包含一些脚本（postinst、postrm、preinst、prerm），dpkg 在处理软件包的不同阶段调用这些脚本。我们可以使用 dpkg -I 命令，显示.deb 软件包存档文件中的这些文件：

```
$ dpkg -I /var/cache/apt/archives/zsh_5.3-1_amd64.deb | head
new debian package, version 2.0.
size 814486 bytes: control archive=2557 bytes.
     838 bytes,    20 lines    control
     3327 bytes,   43 lines    md5sums
     969 bytes,    41 lines    * postinst        #!/bin/sh
     348 bytes,    20 lines    * postrm          #!/bin/sh
     175 bytes,     5 lines    * preinst         #!/bin/sh
     175 bytes,     5 lines    * prerm           #!/bin/sh
Package: zsh
Version: 5.3-1
$ dpkg -I zsh_5.3-1_amd64.deb preinst
#!/bin/sh
set -e
# Automatically added by dh_installdeb
dpkg-maintscript-helper symlink_to_dir /usr/share/doc/zsh zsh-common 5.0.7-3
-- "$@"
# End automatically added section
```

Debian Policy 详细描述了每个文件，指定了调用脚本的时机，以及它们接收到的参数格式。其中处理过程可能会比较复杂，因为如果其中一个脚本失败，dpkg 将通过取消正在进行的安装，或删除操作（只要有可能），尝试返回到令人满意的状态。

dpkg 数据库

你可以在/ var / lib / dpkg /中遍历文件系统上的 dpkg 数据库。该目录包含系统中已安装所有软件包的运行记录。所有已安装软件包的配置脚本都以文件名的形式存储在/ var / lib / dpkg / info /目录下：

```
$ ls /var/lib/dpkg/info/zsh.*
/var/lib/dpkg/info/zsh.list /var/lib/dpkg/info/zsh.md5sums
/var/lib/dpkg/info/zsh.postinst
/var/lib/dpkg/info/zsh.postrm
/var/lib/dpkg/info/zsh.preinst
/var/lib/dpkg/info/zsh.prerm
```

该目录还包含一个用于每个包的.list 文件，该文件包含属于该包的文件列表：

```
$ head /var/lib/dpkg/info/zsh.list
/.
/bin
/bin/zsh
/bin/zsh5
/usr
/usr/lib
/usr/lib/x86_64-linux-gnu
/usr/lib/x86_64-linux-gnu/zsh
/usr/lib/x86_64-linux-gnu/zsh/5.2
/usr/lib/x86_64-linux-gnu/zsh/5.2/zsh
[...]
```

/var/lib/dpkg/status 文件包含了一些数据块（著名的邮件标题的格式，RFC2822），它们描述了每个软件包的状态。control 文件中已安装软件包的信息也会被复制过来。

```
$ more /var/lib/dpkg/status
Package: gnome-characters
Status: install ok installed
Priority: optional
Section: gnome
Installed-Size: 1785
Maintainer: Debian GNOME Maintainers<pkg-gnome-
    ➥ maintainers@lists.alioth.debian.org>
```

```
Architecture: amd64
Version: 3.20.1-1
[...]
```

我们来看配置文件，看它们是如何交互的。一般来说，preinst 脚本在安装包之前执行，而 postinst 在它之后执行。同样，prerm 在移除包之前被调用，postrm 在移除包之后被调用。软件包的更新等同于删除以前的版本并安装新版本。这里不可能详细描述所有可能的场景，但我们将讨论最常见的几个：安装、更新和删除。

这些处理过程显得相当混乱，图形化的表示可能会清晰一些。Manoj Srivastava 用视图来解释 dpkg 如何调用配置脚本。Debian Women 项目也开发了类似的可视化工具。这些工具理解起来还是比较简单的，但都不太全面。

➥https://people.debian.org/~srivasta/MaintainerScripts.html

➥https://wiki.debian.org/MaintainerScripts

注意	本节中使用特定的名称序列命名配置脚本，例如 old-prerm 或 new-postinst。它们分别是旧版本软件包（在更新前已安装）中包含的 prerm 脚本和新版本软件包（由本次更新所安装）中包含的 postinst 脚本。
脚本的符号名称	

安装和升级

以下是在安装（或更新）期间发生的事情。

1. 对于更新，dpkg 会调用 old-prerm upgrade new-version 命令。

2. 还是在更新过程，然后 dpkg 执行 new-preinst upgrade old-version。若首次安装，会调用 new-preinst install 命令。如果包已经被安装并移除了（但未清除，配置文件被保留），dpkg 可能会在最后一个参数中加入老版本号。

3. 解压安装包，如果一个文件存在，就覆盖它，同时临时创建一个副本。

4. 对于更新，dpkg 接下来会调用 old-postrm、upgrade 和 new-version。

5. dpkg 更新所有内部数据（文件列表、配置脚本等），并删除替换文件副本。这个过程是不可逆的，dpkg 无法访问所有之前一个状态必需的元素。

6. dpkg 会更新配置文件（注意不是配置脚本），并询问用户是否能够自动管理安装过程。有关此过程的详细信息，参见 8.4.3 节的介绍。

7. 最后，dpkg 会调用 new-postinst、configure、last-version-configured 来配置软件包。

包移除

下面是在包移除的过程中会发生的事情。

1. dpkg 调用 prerm remove。

2. 除配置文件和配置脚本外，dpkg 删除所有包的文件。

3. dpkg 执行 postrm remove。除了 postrm 以外，其余脚本都会被删除。如果用户没有选择"purge"选项，那么操作将在这一步骤完成。

4. 若是完全移除（dpkg --purge 或 dpkg -p），配置文件也会被删除，连同一些文件副本（*.dpkg-tmp、*.dpkg.old、*.dpkg-new）和一些临时文件一起被删除。随后，dpkg 执行 postrm、purge。

有的时候，软件包会用 debconf 工具，让你输入安装软件时需要的配置信息。上文详细介绍过的 4 个脚本根据 config 脚本收集到的配置信息进行安装。在安装软件时，这个 config 脚本详细定义了 debconf 要问的问题。用户的响应内容被记录在 debconf 数据库中，以供将来参考。在安装软件包之前，这个脚本一般由 apt 来调用，以便在流程开始时将所有问题答案组合在一起。随后 pre-和 post-脚本会使用这些问题的答案，按照用户的意愿进行操作。

debconf 工具 debconf 的出现是为了解决 Debian 中的一个反复出现的问题。在安装任何一个软件包的过程中都无法避免让用户通过 postinst（或者其他类似的脚本）以提问和回答的形式提供必要的配置信息。这就要求在安装或升级期间，用户必须在电脑前以响应各种 postinst 执行期间提问的问题。由于 debconf 的出现，这种麻烦的交互模式终于可以被淘汰了。

debconf 有几个有意思的地方：它要求软件开发人员定义一个用户交互过程；并且其可以把文本信息以本地语言的方式呈现给用户；有不同前端显示问题的方式（字符模式、图形模式、非交互模式）；并支持将这些配置信息集中存储在数据库中，以便共享给其他计算机。它最重要的功能是，可以在启动软件安装或者升级过程前，让用户提前把安装过程需要回答的所有问题都回答完毕。这样用户就可以去处理商务上或者其他方面的工作，而不用守在屏幕前，等着它提示输入相关设置参数。

8.4.3 校验和配置文件

除了前面已经提到的配置脚本和 control 文件之外，Debian 软件包的 control.tar.gz 文件里还包含其他一些文件：

```
# ar p /var/cache/apt/archives/bash_4.4-2_amd64.deb control.tar.gz | tar -tzf -
./
./conffiles
./control
./md5sums
./postinst
./postrm
./preinst
./prerm
```

首先，md5sums 包含了所有文件的 MD5。它的用途是用 debsums 来校验安装包是否被修改过。如果这个文件不存在，dpkg 将在安装时动态生成（并将其存储在 dpkg 数据库中，就像其他控制文件一样）。

conffiles 文件中列出了必须处理的配置文件。管理员可以修改这些配置文件，并且在软件更新时 dpkg 会尝试保留这些更改。

实际上，在这种情况下，dpkg 的行为会尽可能地智能。如果两个版本的配置文件并没有改变，那么 dpkg 就什么也不会做。但是，如果文件已经被修改，它会尝试更新此文件。这会产生两种可能的情况：一是管理员没有碰配置文件，在这种情况下 dpkg 会自动安装新版本；二是文件已经被修改了，在这种情况下 dpkg 会询问管理员希望使用哪个版本（旧版本或者新的）。为了帮助用户作出选择，dpkg 会提供“差异”信息，显示两个版本之间的差异。如果用户选择保留旧版本，新版本将被存储在文件夹的同一个位置并以.dpkg-dist 为后缀名的文件中。如果选择新版本，那么旧版本会被保存在以. dpkg-old 为后缀名的文件中。另外一个可能的操作是由暂时中断 dpkg 来编辑该文件，然后试图重新恢复相关修改（之前通过 diff 验证的差异）。

dpkg 在处理配置文件更新的时候，会跟用户交互，会定期中断其工作，要求管理员输入相关设置参数。这是极为耗时且不方便的。幸运的是，可以让 dpkg 自动响应这些提示。

所以 dpkg 提供了一些选项：—force-confold，保存旧版本配置文件；—force-confnew 使用新版本的配置文件；--force-confdef，告诉 dpkg 让其自主选择（当原始配置文件没有被修改时），如果修改过原始配置文件，则可以使用--force-confnew 或--force-confold。

这些选项都可以在 dpkg 中使用，但管理员大多数时候使用的是 apt-get 或 aptitude。所以有必要了解下它们对应的命令（它们的命令行界面非常相似）。

```
# apt -o DPkg::options::="--force-confdef" -o DPkg::options::="--force-
    ➥ confold" full-upgrade
```

这些选项可以直接存储在 apt 的配置中。做法是将下面这行添加到/etc/apt/apt.conf.d/local 文件中：

```
DPkg::options { "--force-confdef"; "--force-confold"; }
```

这样一来，图形界面的程序也能应用这些配置选项了，例如 aptitude。

你也可以选择让 dpkg 使用交互方式来进行配置：--force-confask 选项要求 dpkg 显示选择配置文件的问题，即使在通常情况下不需要设置也是如此。因此，当重新安装软件包时，dpkg 会针对所有已修改的配置文件向用户进行提问。有时候这种用法会显得很方便，例如配置文件被删除，要重新安装这些原始配置文件，这时候就可以使用--force-confask 选项。因为如果 dpkg 用的是普通模式，它会认为配置文件的移除是一种正确的修改操作，所以不会重新安装用户需要的配置文件。

8.5 小结

在本章中，我们了解了更多关于 Debian 软件包系统的知识。讨论了高级软件包工具（APT）和 dpkg，了解了软件包的基础交互，apt 的高级配置和使用，并深入了解了 Debian 软件包系统、.deb 文件格式，以及 control 文件、配置脚本、校验和及 conffiles 文件。

要点提示：

Debian 软件包是一个软件应用程序的压缩文档。它包含应用程序的文件以及其他元数据，应用程序所需依赖项的名称，以及软件包生命周期的不同阶段（安装、删除和升级）中执行命令的脚本。

与 APT 系列的 apt 和 apt-get 相反，dpkg 工具不知道满足软件包依赖关系的所有可用软件包。因此，要管理 Debian 软件包，需使用 APT 相关工具，因为它们能够自动解决依赖性问题。

可以使用 APT 安装和删除应用程序、更新软件包，甚至升级整个系统。以下是你应该了解的 APT 及其配置的关键点：

- sources.list 文件是定义软件源（或包含软件包的目录）的关键配置文件。
- Debian 和 Kali 使用三个部分，根据每个软件作者选择的许可证区分软件包：main 包是完全符合“Debian 自由软件指南”的所有软件包；“non-free”包含的软件不（完全）符合“自由软件指南”，但可以不受限制地分发；contrib（contributions）

包括了开放源码软件，还有那些无法脱离非自由元素运行的软件。

- Kali 拥有多个存储库，包括：Kali-Rolling（用户的主要存储库），其包含了可安装和最近的软件包；Kali 开发人员使用的 Kali-Dev，并不推荐公众使用；以及 Kali-Bleeding-Edge，通常包含未经测试和未经审核的软件包，它们在提交后不到 24 小时就自动从上游的 Git（或 Subversion）存储库中被构建。
- 使用 APT 时，应首先使用 `apt update` 下载当前可用软件包的列表。
- 可以使用 `apt install package` 把软件添加到系统中。APT 将自动安装必要的依赖关系。
- 要删除软件包，使用 `apt remove package`。它也将删除软件包的反向依赖关系（即依赖于要被移除软件包的软件包）。
- 要删除与包相关的所有数据，可以使用 `apt purge package` 命令“清除”包。与删除不同，这不仅会删除软件包，还会删除其配置文件，有时还会删除关联的用户数据。

我们建议定期升级以安装最新的安全更新。要进行升级，使用 `apt update` 命令，然后再使用 `apt upgrade`、`apt-get upgrade` 或 `aptitude safe-upgrade` 命令。这些命令查找可以升级而不用删除任何已安装软件包的安装包。

对于更重要的升级（例如主要版本升级），需使用 `apt full-upgrade` 命令。通过这个命令，APT 将完成系统升级，即使它必须删除一些过时软件包，或是安装新的依赖关系。这也是你用来定期升级 Kali Rolling 系统的命令。可以回顾一下我们在本章中概述的更新的优缺点。

可以使用几种工具来检查 Debian 软件包：

- `dpkg --listfiles package`（或`-L`）列出了由指定软件包安装的文件。
- `dpkg --search file`（或`-S`）查找包含在参数中传递的文件或路径的所有包。
- `dpkg --list`（或`-l`）显示系统已知的软件包列表及其安装状态。
- `dpkg --contents file.deb`（或`-c`）列出特定`.deb`文件中的所有文件。
- `dpkg --info file.deb`（或`-I`）显示指定的`.deb`文件的标题。
- 各种 `apt-cache` 子命令显示存储在 APT 内部数据库中的大部分信息。

为了避免磁盘使用过多，你应该定期通过`/ var / cache / apt / archives /` 进行排序。可以使用两个命令：`apt clean` 或 `apt-get clean` 完全清空目录；`apt autoclean` 或 `apt-get autoclean` 只删除那些不能再下载的软件包，因为它们已经从镜像中消失了，因此是无用的。

`Aptitude` 是一个交互式程序，可以在控制台上以半图形化的模式使用。这是一个非常

强大的程序，可以帮助你安装和解答软件包的问题。

`synaptic` 是一个图形化的包管理器，具有清晰高效的图形界面。

作为高级用户，你可以在`/etc/apt/apt.conf.d/`中创建文件来配置 APT 的某些方面。你还可以管理软件包的优先级，跟踪自动安装的软件包，一次使用多个分发版或多个体系架构，使用加密签名来验证软件包，并使用本章中概述的技术来升级文件。

尽管 Kali / Debian 的维护者尽了最大努力，但系统升级并不总是像我们希望的那样顺利。发生这种情况时，可以查看 Kali bug 跟踪器和 `https://bugs.debian.org/package` 网站上的 Debian bug 跟踪系统，查看是否已经有人报告了该问题。你也可以尝试降级软件包，或调试并修复之前报错软件的维护脚本。

练习题

练习1——镜像重定向

1. 如果你尝试从 `cdimage.kali.org` 下载 Kali ISO，请找出是哪个镜像点处理你的 ISO 下载请求。
2. 在你的 `sources.list` 文件中添加配置一个源，并更新你的软件包缓存。
3. 将 `kali-bleeding-edge` 存储库添加到你的系统中，并安装 Bleeding-Edge 版本的“set”软件包。

练习2——进一步了解dpkg

1. 找到 `atk6-alive6` 二进制文件的存储路径。
2. 确定安装它的软件包和包中包含的其他文件。
3. 搜索名称中含有“wifi”的本地软件安装包。
4. 在 Kali 软件库中搜索名称中含有“wifi”的工具。
5. 列出由 nmap 软件包安装产生的关联文件。

练习3——dpkg-deb练习

在这个练习中，我们要安装 Nessus。Nessus 安装并不是一个简单的 `apt-get` 命令就能

完成的。具体安装过程如下：

- 从 Nessus 官网下载.deb 安装文件。
- 用 `dpkg` 安装.deb 文件。
- 注册 Nessus 并通过电子邮件获取激活码。
- 在 Kali 中使用 nessuscli 生成一个挑战码。
- 提交挑战码和激活码以访问插件。
- 下载插件并安装。

这是一个漫长的过程，而且整个过程并不简单。用我们刚刚介绍的方法，下载并安装 Nessus，然后创建一个包含插件的软件包并自动安装它们。

1．找到并下载，而后安装并注册 Free Nessus Debian 软件包。

2．不要通过 Nessus Web 界面以编程方式安装插件，而是将插件保存到本地文件。

3．将签名写入 Nessus .deb 软件包，以便可以在安装 Nessus（例如跨多台计算机）时，不必每次都重新下载签名。

4．创建一个能够自动安装插件的新 Nessus.deb 包。

练习4——Kali Multi-Arch多体系架构

这将是一个有趣的练习，因为它非常简单，用一些你已经学过的包处理命令，就能在 Kali 上运行 Windows 程序，这都要归功于 Wine。尽管听起来可能比它实际操作要复杂一些，因为你必须得安装外部体系架构（i386）。

1．使用 `dpkg` 将 32 位体系架构选项添加到你的 Kali 系统里。

2．安装 wine32。

3．使用 Wine 运行 Windows ipscan 程序。

思考题

1．输入 `apt-cache search nmap` 命令时，为什么 `atac` 这个工具也显示在了结果里？

第9章　高级用法

内容

- 修改 Kali 软件包
- 重编译 Linux 内核
- 定制 Kali 自生 ISO 镜像
- 使用 U 盘给自生 ISO 添加持久化功能
- 小结

Kali 已经成为一款高度模块化、定制化的渗透测试框架。从源码级开始，定制化程度可以分为多个级别。所有 Kali 包的源码都是公开可用的。在本章，我们将介绍如何获取、修改包并从中构造自己的定制包。Linux 内核稍微特殊些，我们会用一整节来重点介绍（见 9.2 节），我们会讨论如何获取内核源码，如何配置内核编译选项，如何编译和构建相关的内核包。

第二级别的定制，是构建自生（Live）ISO 镜像。我们将介绍如何使用 live-build 工具提供的大量钩子功能和配置选项，定制生成 ISO 镜像，包括使用定制的 Debian 软件包替代镜像上的软件包。

最后讨论如何在 U 盘上构造一个存储持久的自生 ISO，它会在操作系统重启后，对文件和系统的更改进行保存。

9.1　修改Kali软件包

修改 Kali 软件包，通常是 Kali 贡献者和开发人员的任务。他们使用新的上游版本更新软件包，调整默认配置以便更好地集成到发行版中，或者修复用户报告的 bug。但是我们可能会有官方软件包无法满足的特殊需求，因此我们要知道如何修改软件包。

为什么要修改软件包？我们知道，如果要修改软件中的某一部分，可以在获取它的源码后（通常使用 `git`），直接编译运行修改后的版本。如果只是放在你自己的 Home 目录下运行，这当然没问题。但如果需要在系统范围内安装（比如 `make install` 安装），它会污染你的文件系统，因为 `dpkg` 不能自动识别这些文件，很快就会产生软件包依赖相关的问题。

此外，如果有合适的软件包，还可以很方便地在多台计算机上共享部署，或者在发现软件包没有达到你的预期后，还可以撤销改动。

那么，何时需要修改软件包？先看几个例子：1.假设你是个 SET 工具套件的重度定制用户，你发现了某个新版本，但 Kali 的开发者们都在忙着准备一个会议，而你想要现在就尝试一下，这时就需要自己去动手更新包了。2.假设你正想办法让你的 MIFARE NFC 卡正常工作，你可能需要重编译“libfreefare”包，启用调试消息，以便在 bug 报告中输出更多有用的调试信息。3.假设你的“pyrit”程序挂了，并输出了一条神秘的报错信息。经一番搜索后发现，上游的 Github 仓库中已修复了这个 bug，你此时希望打上这个补丁，重新编译软件包。

我们会在本章后续章节中介绍这些例子。为更好地指导大家将其拓展应用于其他情况，我们会对其进行归纳总结，但不可能涵盖所有情况。所以如果你遇到了问题，请自行去论坛上查找解决方案或寻求帮助（见第 6 章）。

无论做哪种修改，其大体流程基本相同：获取源码包，解压源码包，修改源码，然后重新编译生成。对于每一步，都会有很多工具可用。这里无法介绍所有工具，我们仅挑选其中相关度较高和较常用的工具进行讲解。

9.1.1　获取源码

重编译 Kali 软件包从获取源码开始。一个源码包由多个文件组成：主要文件是`*.dsc`（Debian Source Control）文件，它列出其他附属文件，可以是`*.tar`、`bz2`、`xz`，有时也可以是`*.diff.gz` 或者 `debian.tar.gz`、`bz2`、`xz` 文件。

源码包存在 Kali 镜像源上，可通过 HTTP 方式访问下载。你可以使用浏览器去下载，但最简单的办法是通过 `apt source` source_package_name 命令获得源码包。该命令需要 `/etc/apt/sources.list` 文件里有相关 `deb-src` 的行，且索引文件已更至最新（运行 `apt update` 命令）。正常情况下，Kali 默认是没有添加这一行的，因为很少有 Kali 用户需要源码包，但没关系，添加它也很简单（见 8.1.3 节示例文件及 8.1.2 节的相关介绍）。

```
$ apt source libfreefare
Reading package lists... Done
NOTICE: ’ libfreefare’ packaging is maintained in the ’ Git’ version control
system at: git://anonscm.debian.org/collab-maint/libnfc.git
Please use:
git clone git://anonscm.debian.org/collab-maint/libnfc.git
to retrieve the latest (possibly unreleased) updates to the package.
Need to get 119 kB of source archives.
```

```
Get:1 http://archive-2.kali.org/kali kali-rolling/main libfreefare 0.4.0-2 (dsc)
[2,090 B]
Get:2 http://archive-2.kali.org/kali kali-rolling/main libfreefare 0.4.0-2 (tar)
[113 kB]
Get:3 http://archive-2.kali.org/kali kali-rolling/main libfreefare 0.4.0-2 (diff)
[3,640 B] Fetched 119 kB in 1s (63.4 kB/s)
gpgv: keyblock resource ‘/home/rhertzog/.gnupg/trustedkeys.gpg’ : file open
error
gpgv: Signature made Tue 04 Mar 2014 06:57:36 PM EST using RSA key ID 40AD1FA6
gpgv: Can’ t check signature: public key not found
dpkg-source: warning: failed to verify signature on ./libfreefare_0.4.0-2.dsc
dpkg-source: info: extracting libfreefare in libfreefare-0.4.0
dpkg-source: info: unpacking libfreefare_0.4.0.orig.tar.gz
dpkg-source: info: unpacking libfreefare_0.4.0-2.debian.tar.xz
$ cd libfreefare-0.4.0
$ ls
AUTHORS    CMakeLists.txt   COPYING   HACKING           m4            README
ChangeLog  configure.ac     debian    libfreefare       Makefile.am   test
cmake      contrib          examples  libfreefare.pc.in NEWS          TODO
$ ls debian
changelog     copyright               libfreefare-dev.install  rules
compat        libfreefare0.install    libfreefare-doc.install  source
control       libfreefare-bin.install  README.Source           watch
```

在上面这个例子中，我们从 Kali 镜像源获得源码包，版本号没有“kali”字样，其包名与 Debian 中的一样，这意味着该源码包没有专门针对 Kali 做特别修改。

如果需要一个特定版本的源码包，且在当前的`/etc/apt/sources.list` 源列表中也没找到，那么最简单的办法是，直接在 `http://pkg.kali.org` 中查找它对应的.dsc 文件的 URL。然后，通过 `dget` 命令来处理这个 URL（命令来自 *devscripts* 软件包）。

在 `kali-bleeding-edge` 源中找到 libfreefare 源码包的 URL 后，使用 `dget` 命令下载。首先下载`.dsc` 文件，然后解析查看是否引用了其他文件，有的话，也一起下载。

```
$ dget http://http.kali.org/pool/main/libf/libfreefare/libfreefare_0.4.0+0~
   ➥ git1439352548.ffde4d-1.dsc
dget: retrieving
http://http.kali.org/pool/main/libf/libfreefare/libfreefare_0.4.0+0~
   ➥ git1439352548.ffde4d-1.dsc
% Total % Received % Xferd Average Speed  Time  Time  Time Current
                     Dload  Upload Total Spent Left Speed
100 364 100 364 0 0 852 0 --:--:-- --:--:-- --:--:-- 854
100 1935 100 1935 0 0 2650 0 --:--:-- --:--:-- --:--:-- 19948
```

```
dget: retrieving
http://http.kali.org/pool/main/libf/libfreefare/libfreefare_0.4.0+0~
   ➥ git1439352548.ffde4d.orig.tar.gz
[...]
dget: retrieving
http://http.kali.org/pool/main/libf/libfreefare/libfreefare_0.4.0+0~
   ➥ git1439352548.ffde4d-1.debian.tar.xz
[...]
libfreefare_0.4.0+0~git1439352548.ffde4d-1.dsc:
dscverify: libfreefare_0.4.0+0~git1439352548.ffde4d-1.dsc failed signature
check:
gpg: Signature made Wed Aug 12 06:14:03 2015 CEST
gpg: using RSA key 43EF73F4BD8096DA
gpg: Can’t check signature: No public key
Validation FAILED!!
$ dpkg-source -x libfreefare_0.4.0+0~git1439352548.ffde4d-1.dsc
gpgv: Signature made Wed Aug 12 06:14:03 2015 CEST
gpgv: using RSA key 43EF73F4BD8096DA
gpgv: Can’t check signature: No public key
dpkg-source: warning: failed to verify signature
on ./libfreefare_0.4.0+0~git1439352548
   ➥ .ffde4d-1.dsc
dpkg-source: info: extracting libfreefare in libfreefare-
0.4.0+0~git1439352548.ffde4d
dpkg-source: info: unpacking
libfreefare_0.4.0+0~git1439352548.ffde4d.orig.tar.gz
dpkg-source: info: unpacking libfreefare_0.4.0+0~git1439352548.ffde4d-
1.debian.tar.xz
```

在上面的例子中，因为源码包的 PGP 签名无法验证，`dget` 没有自动下载并解压源码包。因此，我们通过 `dpkg-source -x dsc-file` 命令来提取源码包。当然还可以通过加上`--allow-unauthenticated` 或`-u` 参数，强行提取源码包；或者还可以使用`--download-only` 选项，跳过解压过程，只下载源码包。

从 Git 仓库获取源码

你可能已经从 `apt source` 命令的调用回显中，注意到这个源码包在某个 Git 代码仓库中维护。它可能指向一个 Debian 或者 Kali 的 Git 仓库。

所有Kali特有的软件包，均由位于`git.kali.org`[1]的代码仓库进行维护，所以可以使用`git clone git://git. kali.org/packages/`source-package命令

1 http://git.kali.org

获取源码。如果该操作未得到预期的源码，尝试使用`git checkout kali/master`命令，切换到`kali/master`分支看看。

这跟 `apt source` 获取方式的不同之处在于，通过 `git` 获取的源码，不会自动把补丁打上。可以通过查看 `debian/patches/`目录，了解 Kali 做了哪些改动。

```
$ git clone git://git.kali.org/packages/kali-meta
Cloning into ’ kali-meta’ ...
remote: Counting objects: 760, done.
remote: Compressing objects: 100% (614/614), done.
remote: Total 760 (delta 279), reused 0 (delta 0)
Receiving objects: 100% (760/760), 141.01 KiB | 0
bytes/s,
   ➥ done.
Resolving deltas: 100% (279/279), done.
Checking connectivity... done.
$ cd kali-meta
$ ls
debian
$ ls debian
changelog compat control copyright rules source
```

作为备选方案，可以通过 git 仓库来获得源码，但是，Kali 开发者处理这些代码仓库时，使用的是另一套打包流程，他们使用 `git-buildpackage` 工具套件，这里不对该工具包进行具体讲解，读者可通过以下链接自行了解：

➨ `https://honk.sigxcpu.org/piki/projects/git-buildpackage/`

9.1.2 安装编译依赖包

现在我们有源码包了，下一步需要安装编译依赖。在将源码编译成二进制文件包的过程中，这些依赖包是必需的。

在 `debian/control` 文件中，每个源码包均使用 Build-Depends 字段声明了它的编译依赖。我们通过 `apt` 命令来安装这些依赖包（假设所在的目录下，包含一个已解压的源码包）：

```
$ sudo apt build-dep ./
Note, using directory ’ ./’ to get the build dependencies
Reading package lists... Done
Building dependency tree
```

```
Reading state information... Done
The following NEW packages will be installed:
   autoconf automake autopoint autotools-dev debhelper dh-autoreconf
   dh-strip-nondeterminism gettext intltool-debian libarchive-zip-perl
   libfile-stripnondeterminism-perl libtool po-debconf
0 upgraded, 13 newly installed, 0 to remove and 0 not upgraded.
Need to get 4 456 kB of archives.
After this operation, 14,6 MB of additional disk space will be used.
Do you want to continue? [Y/n]
[...]
```

在该例中，APT 下的软件包即可满足全部的编译依赖。在实际情况中，可能不总是这样，比如 kali-rolling 并不确保可安装这些编译依赖（只考虑到了依赖项目的二进制包）。实际上，二进制依赖包与编译依赖往往是紧密耦合的，绝大多数包相应的编译依赖都是满足的。

9.1.3　修改源码

本节无法覆盖到你想要对某个源码包做出的所有可能修改，这相当于讲解所有Debian打包机制的具体细节 [1]。我们会讲解前面提到的三个常见案例，解释一些无法绕开的部分（比如changelog文件的维护）。

首先要做的是更改软件包的版本号，这样重编译包以后，才能与 Kali 或 Debian 原有包区别开来。通常通过添加一个后缀，标识哪个人或者公司做了这些更改。因为 `buxy` 是笔者的 IRC 昵称，所以笔者用这个作为后缀。这种改动最好通过 devscripts 软件包的 `dch` 命令（Debian CHangelog）来操作，具体命令是 `dch --local buxy`。该命令会调用一个文本编辑器（`sensible-editor`，启动 `VISUAL` 或者 `EDITOR` 环境变量指向的编辑器，否则指向 `/usr/bin/editor`）。在编辑器中写下这次重构所引入的更改。这个编辑器显示了 `dch` 确实对 `debian/changelog` 文件做了修改。

```
$ head -n 1 debian/changelog
libfreefare (0.4.0-2) unstable; urgency=low
$ dch --local buxy
[...]
$ head debian/changelog
libfreefare (0.4.0-2buxy1) UNRELEASED; urgency=medium
```

1 https://www.debian.org/doc/manuals/maint-guide/

```
   * Enable --with-debug configure option.

  -- Raphael Hertzog <buxy@kali.org> Fri, 22 Apr 2016 10:36:00 -0400

libfreefare (0.4.0-2) unstable; urgency=low

   *  Update debian/copyrtight.
      Fix license to LGPL3+.
```

如果经常需要修改，最好分别设置好 DEBFULLNAME 和 DEBEMAIL 这两个环境变量。这两个环境变量以及包括 dch 在内的很多打包工具都会被用到。就像上面这个例子中以“--”开头的行一样，这些打包工具，会把这两个变量中的信息嵌入到跟踪行。

打补丁

在我们的一个案例中，pyrit 源码包已下载，但我们在上游 git 仓库中，发现了一个新的补丁。这种情况很常见，操作起来应该也容易。但是，根据源码包格式和使用的 Git 打包工作流不同，补丁文件的处理方式也是不一样的。

已解压的源码包 需要先运行 apt source pyrit 命令，生成一个类似 pyrit-0.4.0 的目录。此时，可以直接使用 patch -p1 < patch-file 命令进行打补丁操作：

```
$ apt source pyrit
[...]
$ cd pyrit-0.4.0
$ wget https://github.com/JPaulMora/Pyrit/commit/14
   ➥ ec997174b8e8fd20d22b6a97c57e19633f12a0.patch -O /tmp/pyrit-patch
[...]
$ patch -p1 </tmp/pyrit-patch
patching file cpyrit/pckttools.py
Hunk #1 succeeded at 53 (offset -1 lines).
$ dch --local buxy “Apply patch to work with scapy 2.3”
```

此时，已经完成了给源码包打补丁的操作，现在可以编译打补丁后的版本产生二进制文件了（具体操作见 9.1.4 节）。但如果你试着编译这个更新后的源码包，会报错“unexpected upstream changes”。这是因为 pyrit（包括大多数的源码包）使用 3.0（quilt）源码格式（见 debian/source/format 文件），该格式要求对上游代码的任何改动，必须分开记录在 debian/patches/目录下，debian/patches/series 文件表示补丁应用的顺序。可以通过运行 dpkg-source --commit 命令，将所做修改注册为一个新的补丁文件。

```
$ dpkg-source --commit
dpkg-source: info: local changes detected, the modified files are:
  pyrit-0.4.0/cpyrit/pckttools.py
Enter the desired patch name: fix-for-scapy-2.3.patch
dpkg-source: info: local changes have been recorded in a new patch: pyrit-
0.4.0/debian/
   ➥ patches/fix-for-scapy-2.3.patch
$ tail -n 1 debian/patches/series
fix-for-scapy-2.3.patch
```

quilt 补丁系列　这个补丁管理规范，已被 quilt 工具推广使用，因此其同样兼容了“3.0（quilt）”源码包格式，唯一的区别就是它使用 debian/patches 目录替代了 patches 目录。该工具可直接从 APT 同名安装，这里还有个挺不错的使用指南：

➨ https://raphaelhertzog.com/2012/08/08/how-to-use-quilt-to-manage-patches-in-debian-packages/

如果源码包使用 1.0 或 3.0（native）格式，则无须注册对上游代码的改动。这次改动会被自动捆绑到所生成的源码包里去。

Git 仓库管理的源码包　如果使用 Git 来获取源码，情况会更为复杂。当前有多种 Git 工作流及配套工具可供使用。显然，这么多 Debian 软件包，并没有使用同一个 Git 工作流进行管理。上文已讲解了相关源码格式的区别，此外，还需检查源码树中，是否已提前打好了补丁，或者还只是仅仅存储在 `debian/patches` 目录下（这种情况下，补丁会在编译时应用）。

最常用的工具是 git-buildpackage，所有 `git.kali.org` 上的代码仓库均由它来管理。这种方式下，补丁没有预先打在源码上，只是存储在 `debian/patches` 目录下。可以手工在这个目录下添加补丁文件，并在 `debian/patches/series` 文件中列出来。但 git-buildpackage 的用户更倾向于使用 `gbp pq` 将整个补丁序列作为一个单独分支来编辑，这样就可以根据个人喜好，随意扩展或衍合（rebase）。可以查看 `gbp-pq(1)`手册，学习如何使用该命令。

`git-dpm`（命令同名）是另一个常用的 Git 包管理工具。它在 `debian/.git-dpm` 下记录元数据，通过合并从 `debian/patches` 的内容中不断重建的分支，来保持源码树处于打补丁的状态。

调整编译选项

何时需要调整编译选项？当你想要启用一个官方源码包没有激活的功能或特性，或者当

你想要定制化编译参数时，这些参数可能是编译时，`./configure` 的选项参数，也可能是通过环境变量来设置的编译参数。

这种情况下，仅仅需要对 `debian/rules` 做修改即可，`debian/rules` 文件会驱动编译过程中的每个步骤。最简单的情况是，能直观看到关于初始配置（./configure）和实际编译（`$(MAKE)`或 `make`）的行。如果这些命令未被显式调用，它们可能会对另一个显式调用的命令有附加作用，这种情况下，可查阅相应文档，了解如何更改默认行为。使用 `dh` 命令打包的软件包，需要对 `dh_auto_configure` 或 `dh_auto_build` 命令进行覆盖（详情见对应的参考手册，`man pages`）。

为更具体地说明，我们来看一个例子。我们打算修改 libfreefare 包，通过给`./configure` 脚本传递`--enable-debug` 参数开启调试模式，这样可以从 NFC 工具中获取更详细的打印日志，对于无法识别的 Mifare NFC 卡，能获得一个更详细的 bug 报告。因为该包使用 `dh` 来构建编译，需要添加（或者修改）`override_dh_auto_configure`。下面这段是从 libbfreefare 的 `debian/rules` 文件中摘选的：

```
override_dh_auto_configure:
	dh_auto_configure -- --without-cutter --disable-silent-rules --
enable-debug
```

打包新的上游版本

我们还是来看一个例子，假如你经常使用 SET 攻击套件，你发现了一个新的上游版本（7.4.5），该版本还没有及时应用到 Kali 中（此例中，Kali 中的当前版本是 7.4.4）。你想尝尝鲜，自己编译一下更新包。这只是个小版本的迭代，你也不想要在此版本之上做任何修改。

为了更新源码包，你首先从 Kali 上获取了当前老版本的源码包（7.4.4 版本），然后将该版本下的 `debian` 目录整个复制到新的版本中去，最后在 `debian/changelog` 中标记新的版本号。

```
$ apt source set
Reading package lists... Done
NOTICE: 'set' packaging is maintained in the 'Git' version control system at:
git://git.kali.org/packages/set.git
Please use:
git clone git://git.kali.org/packages/set.git
to retrieve the latest (possibly unreleased) updates to the package.
Need to get 42.3 MB of source archives.
[...]
```

```
dpkg-source: warning: failed to verify signature on ./set_7.4.4-0kali1.dsc
dpkg-source: info: extracting set in set-7.4.4
dpkg-source: info: unpacking set_7.4.4.orig.tar.gz
dpkg-source: info: unpacking set_7.4.4-0kali1.debian.tar.xz
dpkg-source: info: applying edit-config-file dpkg-source:
info: applying fix-path-interpreter.patch
$ wget https://github.com/trustedsec/social-engineer-
toolkit/archive/7.4.5.tar.gz -O
   ➥ set_7.4.5.orig.tar.gz
[...]
$ tar xvf set_7.4.5.orig.tar.gz
[...]
social-engineer-toolkit-7.4.5/src/wireless/wifiattack.py
$ cp -a set-7.4.4/debian social-engineer-toolkit-7.4.5/debian
$ cd social-engineer-toolkit-7.4.5
$ dch -v 7.4.5-0buxy1 "New upstream release"
```

如此这般即可，接下来就可以编译更新后的源码包了。

根据上游版本引入的变化的类型，有时可能需要更改编译依赖和运行时依赖，并预装相应文件。这方面仍有许多相关操作，本书就不再赘述了。

9.1.4　开始编译

修改完源码包，接下来即可开始编译生成二进制软件包或`.deb` 文件了。整个过程由`dpkg-buildpackage` 命令管理，大概情况如下所示。

```
$ dpkg-buildpackage -us -uc -b
dpkg-buildpackage: source package libfreefare
dpkg-buildpackage: source version 0.4.0-2buxy1
dpkg-buildpackage: source distribution UNRELEASED
dpkg-buildpackage: source changed by Raphael Hertzog <buxy@kali.org>
dpkg-buildpackage: host architecture amd64
dpkg-source --before-build libfreefare-0.4.0
[...]
   dh_builddeb
dpkg-deb: building package 'libfreefare0' in '../libfreefare0_0.4.0-
2buxy1_amd64.deb'.
dpkg-deb: building package 'libfreefare-dev' in '../libfreefare-dev_0.4.0-
2buxy1_amd64.deb'.
dpkg-deb: building package 'libfreefare-bin' in '../libfreefare-bin_0.4.0-
2buxy1_amd64.deb'.
```

```
dpkg-deb: building package 'libfreefare-doc' in '../libfreefare-doc_0.4.0-
2buxy1_all.deb'.
  dpkg-genchanges -b >../libfreefare_0.4.0-2buxy1_amd64.changes
dpkg-genchanges: binary-only upload (no source code included)
  dpkg-source --after-build libfreefare-0.4.0
dpkg-buildpackage: binary-only upload (no source included)
```

`-us -uc` 选项用于关闭对某些文件（`.dsc`、`.changes`）的签名操作，因为如果没有 `changelog` 文件中对应人的 GnuPG 密钥的话，这个操作就会失败。`-b` 选项表示只编译生成二进制文件。这种情况下，不会创建`.dsc` 文件，只生成二进制文件（`.deb`）。因此，这个选项可以避开一些源码包编译失败的情况，比如没有在补丁管理系统中正确记录所有更改，它就会报错，同时中断编译进程。

如 `dpkg-deb` 的回显所示，所生成的二进制文件包位于父目录下（源码包的父目录），此时即可使用 `dpkg -i` 或 `apt install` 命令来安装它们。

```
$ sudo apt install ../libfreefare0_0.4.0-2buxy1_amd64.deb \
    ../libfreefare-bin_0.4.0-2buxy1_amd64.deb
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting ’ libfreefare0’ instead of ’ ../libfreefare0_0.4.0-
2buxy1_amd64.deb’
Note, selecting ’ libfreefare-bin’ instead of ’ ../libfreefare-bin_0.4.0-
2buxy1_amd64.deb’
The following packages will be upgraded:
libfreefare-bin libfreefare0
2 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Need to get 0 B/69,4 kB of archives.
After this operation, 2 048 B of additional disk space will be used.
[...]
```

推荐使用 `apt install` 命令，与 `dpkg -i` 相比，它的好处在于可以比较优雅地处理所缺少的依赖软件包。但不久之前，你还不得不使用 `dpkg` 命令，因为那时 APT 还无法处理 APT 源之外的单个`.deb` 文件。

dpkg-buildpackage 封装	通常，Debian开发者使用一个更高层次的程序，比如`debuild`。该程序正常运行`dpkg-buildpackage`，同时增加了对`lintian`程序的调用。`lintian`用来运行各种检查，从而验证所生成的软件包是否符合Debian策略 [1]。这个脚本会清除

1 https://www.debian.org/doc/debian-policy/

环境变量，因此本地环境变量无法污染到编译过程。`debuild`命令是devscripts套件中的工具之一，与其他工具共享配置文件，以简化维护者的工作。

9.2　重编译Linux内核

Kali 所提供的内核，已包含尽可能多的功能和驱动程序，从而尽可能覆盖到种类繁多的硬件配置。这也是很多人选择重新编译内核的原因，只留下确实需要的部分。选择重编译内核，主要有两点考虑：一是内存优化。只要是内核代码，不管它是否使用，均会占用物理内存。因为内核静态编译的部分，任何时候都不会移到交换空间里去。这些从不使用的驱动程序和自带功能，会导致系统性能的整体下降。二是安全考虑。减少不必要的驱动程序和内核功能，只有内核中需要的部分在运行，可降低计算机安全风险。

注意　如果选择自行编译内核，你需接受该事实：Kali 不向定制的内核提供安全更新包。如坚持使用 Kali 原生内核，你会受益于 Debian Project 组织提供的更新服务。

如果想使用补丁代码中的特定功能（且标准内核版本中没有），此时同样需要重编译内核。

Debian 内核手册　Debian 内核小组维护了一个 Debian 内核手册（对应有个 debian-kernel-handbook 软件包），该手册全面记录了内核相关的工作，以及如何处理 Debian 官方的内核包。如果不满足于本节所提供的信息，可通过下面这个链接，查找相关资料：

➨ `http://kernel-handbook.alioth.debian.org`

9.2.1　简介与准备工作

Debian 和 Kali 均以软件包的形式管理内核，而不是传统的编译和安装内核。因内核受软件包管理系统控制，所以它可以被干净地卸载，也可以同时部署于多台机器之上。此外，与内核包相关的脚本，可自动与引导加载程序（bootloader）和 initrd 生成器进行交互。

虽然上游 Linux 源码包含编译 Debian 内核包的一切所需，但仍需要安装 build-essential 软件包。此外，内核的配置还需要安装 libncurses5-dev 软件包。最后，需要安装 fakeroot 软件包，用于在非管理员权限下创建 Debian 包。

```
# apt install build-essential libncurses5-dev fakeroot
```

9.2.2 获取源码

因为 Linux 内核源码也有软件包格式，所以可直接通过安装相应的 linux-source-version 来获取源码包。`apt-cache search ^linux-source` 命令可列出 Kali 最新内核源码包。

注意这些包里的源码，跟Linus Torvalds和其他内核开发者发布的Linux源码[1]并不会完全一致。所有Linux发行版都是这样，包括Debian和Kali，都会打上一堆补丁，这些补丁有些可能在上游版本里压根就找不到。这些补丁的内容包括：来自新版本中的漏洞修复、新增功能特性及驱动程序的反向移植；尚未合并到上游Linux kernel源码树的新功能；Debian或Kali自己所做的定制修改等。

接下来我们来看一个 Linux kernel 4.9 版本的示例，其原理过程同样适用于其他内核版本。

在该例中，假设 linux-source-4.9 的二进制包已安装。注意，我们安装包含上游源码的二进制包，但不获取名为 linux 的 Kali 源码包。

```
# apt install linux-source-4.9
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  bc libreadline7
Suggested packages:
  libncurses-dev | ncurses-dev libqt4-dev
The following NEW packages will be installed:
  bc libreadline7 linux-source-4.9
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
Need to get 95.4 MB of archives.
After this operation, 95.8 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
[...]
# ls /usr/src
linux-config-4.9 linux-patch-4.9-rt.patch.xz linux-source-4.9.tar.xz
```

注意，包中含有`/usr/src/linux-source-4.9.tar.xz` 文件，它是内核源码的压缩包。需要先把它解压到新目录里去，比如`~/kernel/`目录下（无须在`/usr/src/`下，编译内核不需要特殊权限）。

1 https://kernel.org/

```
$ mkdir ~/kernel; cd ~/kernel
$ tar -xaf /usr/src/linux-source-4.9.tar.xz
```

9.2.3　配置内核

接下来就是根据自己的情况，对内核进行配置。具体配置什么，取决于你的目的。

内核的编译工作依赖于一个内核配置文件。大多数情况下，都希望配置文件与 Kali 的配置文件类似，该文件一般处于`/boot` 目录下。此时，无须从头配置所有内容，在`/boot/config-version` 的基础上配置即可（使用 `uname -r` 命令，即可获得版本号）。复制该文件至内核源码目录下，重命名为或替换原有`.config` 文件即可。

```
$ cp /boot/config-4.9.0-kali1-amd64 ~/kernel/linux-source-4.9/.config
```

另外，内核本身还提供了默认配置文件，位于 `arch/arch/configs/*_defconfig`，可直接使用 `make x86_64_defconfig`（64 位 PC）或 `make i386_defconfig`（32 位 PC）来选择内核配置文件。

如果对内核配置文件没有改动需求，到这里就结束了，可直接跳过本节以下部分，至 9.2.4 节。如果需要改配置文件或者完全从头配置，那就需要花点时间。内核源码目录下有多种专用界面，这些界面被 `make target` 命令所调用，`target` 可以是下列所述之一。

`make menuconfig` 会启动一个文本模式的内核配置界面（需安装 libncurses5-dev 软件包），该界面以分层分类的形式导航。空格键用于改变选择状态；回车键执行屏幕底部的所选按钮，Select 进入当前选择的子目录；Exit 关闭当前界面或返回上一级目录；Help 用于显示所选择项的详细信息；方向键用于在选项或按钮之间移动。在主界面选择 Exit，可退出配置程序，同时程序会提醒你保存改动的内容。

其他界面也有类似功能，但通常有个现代化的图形界面，比如 `make xconfig` 使用 Qt 图形界面；`make gconfig` 使用 GTK+。前者需要安装 libqt4-dev，后者需要安装 libglade2-dev 和 libgtk2.0-dev。

处理旧的.config 文件　如果手上有个老版本内核的`.config` 文件，在编译新内核时，要先更新它。`make oldconfig` 会交互式地询问新的配置选项，逐个手动回答确认；`make olddefconfig` 对这些询问直接采用默认答案；`make oldnoconfig` 不对新选项做选择。

9.2.4 编译及构建内核包

重编译前清理环境 如果在某个目录里已编译过内核，但需要再从头构建所有内容（比如大幅改动了内核配置），首先应先运行 `make clean` 来移除所有编译生成的文件。`make distclean` 会删除更多的生成文件，包括`.config` 文件，所以进行这个操作之前，请先备份。

内核配置准备完毕后，通过运行 `make deb-pkg` 命令，即可生成以下 5 个标准 Debian 包，这些包均是`.deb` 格式：

- linux-image-version，包含内核镜像和相应模块。
- linux-headers-version，包含编译构建外部模块的头文件。
- linux-firmware-image-version，包含某些驱动所需的固件文件（如从 Debian 或 Kali 的源码包编译，可能不含该文件）。
- linux-image-version-dbg，包含内核镜像及其模块的调试符号。
- linux-libc-dev，包含某些用户空间程序库相关的头文件，如 GNU 的 C 语言库（glibc）。

版本号由上游版本号（如 Makefile 中所定义的 `VERSION`、`PATCHLEVEL`、`SUBLEVEL` 和 `EXTRAVERSION` 变量）和 `LOCALVERSION` 配置参数，以及 `LOCALVERSION` 环境变量连接而成。包的版本号重用相同的字符串，同时附加一个递增的版次号（存储在`.version` 文件内），但可使用 `KDEB_PKGVERSION` 环境变量覆盖。

```
$ make deb-pkg LOCALVERSION=-custom KDEB_PKGVERSION=$(make kernelversion)-1
[...]
$ ls ../*.deb
../linux-headers-4.9.0-kali1-custom_4.9.2-1_amd64.deb
../linux-image-4.9.0-kali1-custom_4.9.2-1_amd64.deb
../linux-image-4.9.0-kali1-custom-dbg_4.9.2-1_amd64.deb
../linux-libc-dev_4.9.2-1_amd64.deb
```

为实际使用所编译生成的内核，剩下的步骤就是使用 `dpkg -i file.deb` 命令安装所需内核包。“linux-image”包是必须要安装的；当需要编译外部内核模块时，才需要安装“linux-headers”包，比如存在`"*-dkms"`包需要安装时（可使用 `dpkg -l "*-dkms" | grep ^ii` 命令检查）。其他包文件一般无须安装（除非你知道为什么需要安装它们）。

9.3　定制Kali自生ISO镜像

Kali具有丰富的功能和极佳的灵活性，只要手持Kali，随时可以炫技，而你仅需一些指引、创新、耐心与实战。同样可以对Kali Linux进行定制，以包含指定文件，执行特定功能（增强性能或增添功能）。Kali ISO of Doom[1]和Kali Evil Wireless Access Point[2]就是两个基于Kali Linux定制的优质项目。接下来，我们一起看看，如何来定制Kali Linux ISO 镜像。

Kali官方的ISO镜像使用live-build[3]构建，live-build由一套脚本组成，可实现ISO镜像创建过程中各个方面的定制和完全自动化。live-build套件使用整个目录结构作为配置文件的输入。我们将该配置文件和相关辅助脚本放在一个`live-build-config`的Git仓库中。接下来，我们使用该仓库作为定制镜像的基础。

本节所使用的命令均运行在最新的 Kali Linux 系统上，如果你使用非 Kali 系统或旧版本 Kali，可能会导致命令执行失败。

9.3.1　准备工作

第一步，安装相关必需的包，获取 live-build 的配置文件。

```
# apt install curl git live-build
[...]
# git clone git://git.kali.org/live-build-config.git
[...]
# cd live-build-config
# ls
auto build_all.sh build.sh kali-config README
```

此时即可通过执行`./build.sh --version`创建一个最新的 Kali ISO 镜像（但没修改过）。整个编译构建时间会非常长，因为它需要现场下载所有文件。编译构建结束后，可以在`images`目录下找到新生成的 ISO 镜像。

1 https://www.offensive-security.com/kali-linux/kali-linux-iso-of-doom
2 https://www.offensive-security.com/kali-linux/kali-linux-evil-wireless-access-point/
3 http://debian-live.alioth.debian.org/live-build/

9.3.2 给自生镜像换个桌面环境

live-build 的 `build.sh` 脚本负责设置生成所需的 `config` 目录。取决于`--variant` 参数的值，可生成不同配置文件。

该脚本通过合并 `kali-config/common` 目录和 `kali-config/variant-X` 目录下的文件生成 `config` 目录，其中 `X` 是`--variant` 参数所给定的变量名。

`kali-config` 目录下有几个常用的桌面环境。

- `e17`：Enlightenment 桌面环境
- `gnome`：GNOME 桌面环境
- `i3wm`：i3 窗口管理器
- `kde`：KDE 桌面环境
- `lxde`：LXDE 桌面环境
- `mate`：Mate 桌面环境
- `xfce`：XFCE 桌面环境

`light`变量特殊一些，它基于XFCE[1]桌面环境，用于生成官方的“light”版ISO镜像，该镜像中的应用程序被做了一定程度的精简。

可使用下面这条命令，轻松创建一个 KDE 桌面环境的 Kali 自生镜像。

```
# ./build.sh --variant kde --verbose
```

`variant`可提供一些更高层次预定义的定制方式，读者可自行查阅Debian Live System的官方手册 [2]，了解很多其他定制方式，而且仅需更改`kali-config`下的子目录内容，即可完成相应定制工作。后续章节会有一些这方面的例子。

9.3.3 更改软件包集合

一旦开始编译构建，`live-build` 会安装所有 `package-lists/*.list.chroot` 下的软件包。我们提供的默认配置在 `package-lists/kali.list.chroot` 文件内，应用配置使用 kali-linux-full（包含全部 Kali 包）。可自行注释这个包，选择其他元包配置或者引入一个更精细化的包集合。还可以通过自由组合元包并补充其他包的方式，来满足个性化需求。

1 https://www.xfce.org/

2 http://debian-live.alioth.debian.org/live-manual/unstable/manual/html/live-manual.en.html

使用 package-lists，仅可引入 Kali 官方源中已有的软件包。如有定制软件包，可以将相应的.deb 文件放置在 packages.chroot 目录下（如使用 GNOME 参数变量，对应路径为 kali-config/config-gnome/packages.chroot）。

元包本身是空的，它的作用是包含众多其他软件包。如果想要同时安装一套软件包，可以使用这种方式，把它们组合在一起。kali-meta 源码包会构建所有 Kali Linux 的元包。

- kali-linux：基础软件系统（被其他元包所包含）
- kali-linux-full：Kali Linux 默认安装的包
- kali-linux-all：Kali 官方能提供的所有包（体积会非常大）
- kali-linux-sdr：软件定义无线电（SDR）工具集
- kali-linux-gpu：GPU 工具集（可利用显卡计算能力的各类工具）
- kali-linux-wireless：无线安全评估与分析套件
- kali-linux-web：Web 应用安全评估工具集
- kali-linux-forensic：取证工具集（现场电子取证）
- kali-linux-voip：VoIP（Voice Over IP）工具集
- kali-linux-pwtools：密码破译工具集
- kali-linux-top10：前十最常用工具
- kali-linux-rfid：RFID 工具集

如要为 live-build 定制软件包集合，可以参照以上这些元包。所有元包及其对应的具体软件清单，可通过单击该链接查看（http://tools.kali.org/kali-metapackages）。

Debconf 预置安装包　可以使用 Debconf 预置文件 preseed/*.cfg 配置需要预装哪些软件包。

9.3.4　使用钩子调整镜像内容

live-build 可以在编译构建的每一步使用钩子。Chroot 钩子是一些可执行脚本，它们使用 chroot 命令，安装目录树下 hooks/live/*.chroot 文件中的所有内容。chroot 命令可临时将系统 root 目录切换到用户所指定的目录，也可拓展用于将整个文件系统存放于指定目录。live-build 在这里就是这么用的，chroot 目录就是预备给自生文件系统（Live Filesystem）的目录。因为应用程序在 chroot 里运行，感知不到目录之外的内容，chroot 钩子也是这样，只可使用和更改 chroot 环境内的任何可用内容。我们依赖这些钩子来进行多个 Kali 特有的功能定制（参阅 kali-config/common/hooks/live/kali-hacks.chroot）。

二进制钩子（hooks/live/*.binary）在构建进程结束之前，在其上下文中执行（没

有 chroot）。你可以更改 ISO 镜像的内容，但此时无法修改自生文件系统的内容，自生文件系统此时已经生成。使用这个特性，可以对 live-build 所生成的 isolinux 配置文件再做些修改。例如，请参阅 kali-config/common/hooks/live/persistence.binary 文件，在其中添加启用持久性的启动菜单条目。

9.3.5 在ISO镜像或自生文件系统中添加文件

另一种非常通用的定制，是在 ISO 镜像或者自生文件系统中添加文件。

把需要添加的文件，放到 includes.chroot 配置目录下的预期位置，比如 kali-config/common/includes.chroot/usr/lib/live/config/0031-root-password 文件，会被放到自生文件系统内对应的/usr/lib/live/config/0031-root-password 文件中。

Live-Boot 钩子 安装在/lib/live/config/XXXX-name 的脚本，由 live-boot 软件包的初始化脚本所执行。脚本会重新配置系统的多个参数，以适配自生系统。你可自行添加脚本，以在运行时定制自生系统。常见的用法是用它实现定制化引导参数。

把需要添加的文件，放到 includes.binary 配置目录下的预期位置，比如 kali-config/common/includes.binary/isolinux/splash.png 文件，会对应覆盖 isolinux 启动加载器的背景图像（对应 ISO 镜像文件系统的/isolinux/splash.png 文件）。

9.4 使用U盘启用自生ISO的持久化功能

9.4.1 持久化特性

下面，我们将讨论如何给 Kali U 盘添加持久化功能。自生系统（Live System）的活动周期是短暂的，退出重启后会恢复到原本的状态，所有系统上存储的数据以及所做的更改，均会丢失。此时，可使用 live-boot 的持久化特性来补救，开启持久化特性的启动参数包含 persistence 关键字。

修改启动引导菜单不是个小任务，Kali 默认提供两个菜单条目来开启持久化功能：Live USB Persistence 和 Live USB Encrypted Persistence，如图 9.1 所示。

图9.1　持久化功能的启动菜单

一旦此功能启用，live-boot 会扫描所有分区，查找标记为 `persistence` 的分区（可被 `persistence-label=value` 启动参数覆盖），在分区内找到对应 `persistence.conf` 文件内的目录（每个目录一行），安装程序会设置持久化功能。特殊值`/union` 会使用一个 `union` 挂载，启用所有目录的全部持久化功能，与基础文件系统对比之后，只会存储所做的修改部分。对应 `persistence.conf` 的持久化目录中的数据，会被存储在文件系统中。

9.4.2　在U盘上设置非加密的持久化功能

本节，假设你已按照 2.1.4 节中的指导，准备好了一个 Kali 自生 U 盘，而且你所持的 U 盘存储容量足够大，可装下整个 ISO 镜像（大概 3GB 以上）和希望持久化的数据。同时，还假设 Linux 识别你的 U 盘为`/dev/sdb`，且其下只包含默认 ISO 镜像的两个分区（分别是 `/dev/sdb1` 和`/dev/sdb2`）。做这些操作的时候，务必非常仔细，一旦分区弄错了磁盘，可能会导致丢失一些重要数据。

添加新分区之前，要先明确镜像的大小，这样可使新分区紧挨着自生镜像。然后使用 `parted` 命令具体创建分区。以下命令分析了名为 `kali-linux-2016.1-amd64.iso` 的 ISO 镜像，假设它也在 U 盘中准备好了。

```
# parted /dev/sdb print
```

```
Model: SanDisk Cruzer Edge (scsi)
Disk /dev/sdb: 32,0GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type     File system Flags
 1      32,8kB  2852MB  2852MB  primary              boot, hidden
 2      2852MB  2945MB  93,4MB  primary
# start=$(du --block-size=1MB kali-linux-2016.1-amd64.iso | awk '{print $1}')
# echo ”Size of image is $start MB”
Size of image is 2946 MB
# parted -a optimal /dev/sdb mkpart primary ”${start}MB” 100%
Information: You may need to update /etc/fstab.

# parted /dev/sdb print
Model: SanDisk Cruzer Edge (scsi)
Disk /dev/sdb: 32,0GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type     File system Flags
 1      32,8kB  2852MB  2852MB  primary              boot, hidden
 2      2852MB  2945MB  93,4MB  primary
 3      2946MB  32,0GB  29,1GB  primary
```

如上所示，/dev/sdb3 分区已创建，下面要使用 mkfs.ext4 命令，将其格式化为 ext4 文件系统，并标记为 persistence（使用 mkfs.ext4 的-L 参数项）。然后将分区挂载到/mnt 目录下，并添加所需的 persistence.conf 配置文件。同样，注意格式化磁盘的过程，一旦失误格错了磁盘，同样会导致丢失重要数据。

```
# mkfs.ext4 -L persistence /dev/sdb3
mke2fs 1.43-WIP (15-Mar-2016)
Creating filesystem with 7096832 4k blocks and 1777664 inodes
Filesystem UUID: dede20c4-5239-479a-b115-96561ac857b6
Superblock backups stored on blocks:
       32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632,
       2654208, 4096000

Allocating group tables: done
Writing inode tables: done
```

```
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done
# mount /dev/sdb3 /mnt
# echo "/ union" >/mnt/persistence.conf
# ls -l /mnt
total 20
drwx------ 2 root root  16384 May 10 13:31 lost+found
-rw-r--r-- 1 root root     8 May 10 13:34 persistence.conf
# umount /mnt
```

如此这般，具有持久化功能的 U 盘就搞定了，可通过启动引导菜单的 Live USB Persistence 选项进入。

9.4.3　在U盘上设置加密的持久化功能

live-boot 同样可以在加密分区上处理持久化文件系统。可通过创建一个 LUKS 加密分区，来存放持久化数据，从而对数据进行保护。

初始步骤与前面类似，但这里先不将其格式化为 ext4 文件系统，而是使用 cryptsetup 命令，将其初始化为一个 LUKS 容器。然后打开容器，并将其设置为 ext4 文件系统。跟前文不一样的地方是，这里不使用/dev/sdb3 分区，而是使用 cryptsetup 创建出来的虚拟分区。虚拟分区表示加密分区解密的内容，它处于/dev/mapper/目录下。如下所示，假设虚拟分区的名字叫 kali_persistence。同样，再次强调，在进行分区操作时务必小心，再三确认无误后再执行。

```
# cryptsetup --verbose --verify-passphrase luksFormat /dev/sdb3

WARNING!
========
This will overwrite data on /dev/sdb3 irrevocably.

Are you sure? (Type uppercase yes): YES
Enter passphrase:
Verify passphrase:
Command successful.
# cryptsetup luksOpen /dev/sdb3 kali_persistence
Enter passphrase for /dev/sdb3:
# mkfs.ext4 -L persistence /dev/mapper/kali_persistence
mke2fs 1.43-WIP (15-Mar-2016)
Creating filesystem with 7096320 4k blocks and 1774192 inodes
```

```
Filesystem UUID: 287892c1-00bb-43cb-b513-81cc9e6fa72b
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632,
        2654208, 4096000

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done

# mount /dev/mapper/kali_persistence /mnt
# echo "/ union" >/mnt/persistence.conf
# umount /mnt
# cryptsetup luksClose /dev/mapper/kali_persistence
```

9.4.4 使用多个持久化存储

如果你的 Kali 自生系统（Live System）有多种使用场景，可能需要使用多个文件系统来满足使用需求，多个文件系统使用不同的标签标记，并在启动菜单中有相应菜单条目与之对应，这些可使用 `persistence-label=label` 启动参数来设定。

假设你是一个渗透测试专家，为客户工作时，你可能会使用加密的持久化分区来保护自己的机密数据，以防 U 盘丢失或被盗用。同时，又希望在同一个 U 盘里存储一些 Kali 展示案例或一些宣传资料。你肯定不想每次启动时，都改一下启动参数，所以可以定制一个自生镜像，在启动菜单里，有各自专用的启动项入口。

第一步是编译定制的自生 ISO 镜像（如 9.3 节所示），尤其是按照 9.3.4 节所述，使用钩子来更改镜像内容。主要步骤是修改 `kali-config/common/hooks/live/persistence-menu.binary` 钩子脚本，使其看起来如下所示（注意 `persistence-label` 参数）：

```
#!/bin/sh


if [ ! -d isolinux ]; then
    cd binary
fi

cat >>isolinux/live.cfg <<END

label live-demo
   menu label ^Live USB with Demo Data
```

```
    linux /live/vmlinuz
    initrd /live/initrd.img
    append boot=live username=root hostname=kali persistence-label=demo
persistence

label live-work
    menu label ^Live USB with Work Data
    linux /live/vmlinuz
    initrd /live/initrd.img
    append boot=live username=root hostname=kali persistence-label=work
       ➥ persistence-encryption=luks persistence
END
```

下一步，编译定制的 ISO，并将其复制到 U 盘。然后创建并初始化两个分区及相应文件系统，以便后续用于持久化存储。第一个分区是非加密的（标记为“demo”），第二个分区是加密的（标记为“work”）。假设/dev/sdb 是我们的 U 盘，而我们所定制 ISO 镜像的大小为 3000MB，那么整个过程执行的命令大概看起来如下所示。

```
# parted /dev/sdb mkpart primary 3000 MB 55%
# parted /dev/sdb mkpart primary 55% 100%
# mkfs.ext4 -L demo /dev/sdb3
[...]
# mount /dev/sdb3 /mnt
# echo “/ union” >/mnt/persistence.conf
# umount /mnt
# cryptsetup --verbose --verify-passphrase luksFormat /dev/sdb4
[...]
# cryptsetup luksOpen /dev/sdb4 kali_persistence
[...]
# mkfs.ext4 -L work /dev/mapper/kali_persistence
[...]
# mount /dev/mapper/kali_persistence /mnt
# echo “/ union” >/mnt/persistence.conf
# umount /mnt
# cryptsetup luksClose /dev/mapper/kali_persistence
```

到这里就全部搞定了，可以在启动 U 盘时，从启动菜单中选择所需的启动项。

为提高安全性添加一个核密码（Nuke Password）

Kali 通过修改 cryptsetup 命令，实现了一个新功能：设置一个核密码（nuke password），在使用该密码时，可以破坏管理加密分区的所有密钥。

这个密码在某些情况下会比较有用：假设你经常出差，需要有个快速的方

法，确保你的数据无法被恢复。启动时，键入核密码（nuke password），而不是真正密码，然后任何人（包括你自己）都无法再访问你的数据。

在使用该功能之前，最好对你的加密密钥做个备份，放到某个安全的地方。

使用下列命令，即可添加一个核密码（nuke password）：

```
# cryptsetup luksAddNuke /dev/sdb4
Enter any existing passphrase:
Enter new passphrase for key slot:
Verify passphrase:
```

该功能更多信息，请猛击下面这个链接。

➨ https://www.kali.org/tutorials/nuke-kali-linux-luks/

9.5 小结

本章，我们学习了如何修改 Kali 源码包，这是将所有应用程序移植到 Kali 的基本构件。我们还探索了如何定制并安装 Kali 内核。然后，讨论了 live-build，以及如何定制一个 Kali Linux ISO。同时，还演示了如何创建加密和非加密的 Kali U 盘。

9.5.1 修改Kali包的小结与建议

修改 Kali 包通常是 Kali 贡献者和开发者的工作，但当你有特殊需求，官方包也无法满足时，知道如何修改包还是相当有用的，尤其是在想要共享自己的修改，在内部进行部署，或者将软件干净地回滚到之前的版本状态之时。

当你需要修改某个软件时，你可能会先下载源码，修改源码，最后使用修改后的版本。但是，如果你的应用程序需要系统范围的安装（比如使用 `make install` 命令安装），那么会导致有一些 `dpkg` 无法识别的文件，污染你的文件系统，从而很快就会产生一些软件包依赖相关的问题。而且，这种类型的软件更改，也很难共享出去。

当创建一个修改版的包时，通常过程是相同的：获取、解压、修改、编译源码包。每一步的任务，都有多个工具可供选择。

开始重编译一个 Kali 软件包时，首先下载其源码包，源码包内含一个`*.dsc`（Debian Source Control）文件，以及该 `dsc` 文件所引用的其他文件。

源码包镜像，可使用 HTTP 访问。最具效率的获取方式是使用 `apt source source-package-name` 命令，使用该命令之前，需要在`/etc/apt/source.list` 文件内添加相关

deb-src 行，并使用 apt update 更新索引文件。

另外，还可使用dget命令（来源于devscripts包）去下载.dsc及其附属文件。Kali专用的包，其源码存放在 git.kali.org [1] 的 Git 仓库中，可使用 git clone git://git.kali.org/packages/source-package 命令获取源码（使用 git checkout kali/master命令切换到kali/master分支）。

源码下载后，使用 sudo apt build-dep ./.命令安装编译依赖，该命令需在软件包的源码目录下运行。

更新源码包步骤如下：

- 首先需要更改版本号，以示与原版本的区别。可使用 dch --local version-identifier 命令，或者使用 dch 命令更改包的其他具体细节内容。
- 使用 patch -p1 < patch-file 或更改 quilt 的补丁系列来完成打补丁的操作。
- 调整编译构建选项，通常在 debian/rules 文件或 debian/目录下的其他文件内可找到这些选项。

更改源码包后，可在源码目录下，使用 dpkg-buildpackage -us -uc -b 命令，生成未签名的二进制包。然后使用 dpkg -i package-name_version_arch.deb 命令安装。

9.5.2　重编译Linux内核的小结与建议

作为一个资深用户，有时候可能需要重编译内核。标准的 Kali 内核加载了太多的功能和驱动，你可能想给标准的 Kali 内核减肥，或者增加一个功能或非标准的驱动，或者给内核打个补丁。请注意，内核误配置会破坏系统稳定性，同时要能接受定制的内核无法使用 Kali 官方提供的安全更新的事实。

大多数内核在定制之前，需要先运行 apt install build-essential libncurses5-dev fakeroot 命令安装一些必需的包。

apt-cache search ^linux-source 命令可列出 Kali 的最新内核版本，apt install linux-source-version-number 会在/usr/src 目录下安装一个内核源码压缩包。

可使用 tar -xaf 命令，将内核源码文件解压到/usr/src 之外的其他目录（比如 ~/kernel 目录）。

配置内核时，请牢记以下几点：

1 http://git.kali.org

- 除非你是资深用户，否则应先填充一个内核配置文件。可通过复制/boot/config-version-string 文件到 ~/kernel/linux-source-version-number/.config，来直接借用 Kali 的标准内核配置文件。或者使用 make architecture_defconfig 命令，根据内核架构，生成一个合适的配置文件。
- 基于文本的 make menuconfig 内核配置工具，会读取.config 文件，然后在一个可导航的菜单中，呈现所有的配置条目。选择一个条目，可显示其文档，显示选项可能的值，以及允许键入新的选项值。

当你在内核源码目录下运行 make clean 命令时，会清除之前编译产生的所有文件。make deb-pkg 命令会生成大概 5 个 Debian 包。其中 linux-image-version.deb 文件包含内核镜像和相应的模块。

使用 dpkg -i file.deb 命令，安装所编译的内核。“linux-image”包是必须要安装的。如果需要编译额外的内核模块（比如有一些“*-dkms”包已安装，使用 dpkg -l "*-dkms" | grep ^ii 命令检查），则需要安装“linux-headers”包。其他包通常不需要安装，除非你真的需要那些包。

9.5.3 定制Kali自生ISO镜像的小结与建议

官方的Kali ISO镜像由live-build[1]工具构建，该工具由一套脚本组成，可实现ISO镜像创建的完全自动化与定制化。

使用 live-build 之前，确保你的 Kali 系统是最新的。

Kali 的 live-build 配置文件，可使用以下两个命令从 Kali 的 Git 仓库中获取：首先使用 apt install curl git live-build 命令，然后再运行 git clone git://git.kali.org/live-build-config.git 命令。

直接运行./build.sh --verbose 命令，可生成一个最新的、未做更改的 Kali ISO 镜像。因为编译过程中需要下载所需包文件，故编译时间会相当长。一旦编译构建完成，可以在 images 目录下找到新生成的 ISO 镜像。如果命令行附带--variant variant 参数变量，会生成给定变量版本的 Kali ISO 镜像。不同变量的定义，对应其配置目录 kali-config/variant-*。主镜像是 gnome 版本的。

通过更改 live-build 的配置目录，可以以多种方式定制 ISO 镜像：

- 可通过修改 package-lists/*.list.chroot 文件，往自生 ISO 镜像中添加或移

1 http://debian-live.alioth.debian.org/live-build/

除包。

- 可通过在 `packages.chroot` 目录下放置相应的`.deb` 文件，将定制的私有包引入自生镜像中去。这些私有包的安装，可通过 `preseed/*.cfg` 文件执行。
- 可通过在 `includes.chroot` 配置目录下预期位置，放置相应文件，从而实现往自生文件系统中添加对应文件的目的。
- 可通过 `hooks/live/*.chroot` 钩子文件，在自生系统的安装过程中执行所需脚本。还可以在所生成自生镜像的启动阶段，执行所需脚本文件。需要通过依赖 `includes.chroot` 配置目录，安装`/usr/lib/live/config/XXXX-name` 文件实现。
- Debian Live System手册 [1]是关于live-build配置和测试的极佳参考文档。

在 U 盘上设置加密和非加密持久化存储。创建标准的 Kali 自生 U 盘相当简单，虽然这个过程看起来复杂，但将非加密和加密的持久化存储功能，添加到便携式设备，以拓展其功能的做法，还是相对比较直观易懂的。

下一章，我们将讨论如何将 Kali 扩展到企业级应用。我们将会讨论如何同时面对数千台机器，对 Kali Linux 做配置管理和扩展定制工作。

练习题

练习1——Fork Kali软件包

1. 请 fork kali-meta 软件包。
2. 请任意选择 3 款工具，包含为新的元软件包。
3. 完成二进制 `deb` 文件的创建，以便于后续使用。

练习2——更新Kali软件包

1. 请为 Kali 系统，打包最新的上游版本 SET 工具套件。
2. 完成二进制 `deb` 文件的创建，以便于后续使用。
3. 你是否可以采取同样的方法，更新 aircrack-ng 软件包？

1 http://debian-live.alioth.debian.org/live-manual/unstable/manual/html/live-manual.en.html

练习3——重编译构建Linux内核

Kali 内核编译，采取了“一体通用”的理念，以最大化支持各类硬件。

1．请安装 likwid 性能评估工具，并启动一次快速的 likwid-bench 基准测试。

2．请安装 graysky2 的内核 GCC 补丁，将额外的 CPU 选项添加到 Linux 内核中。

3．请在完成添加补丁并选择自己的 CPU 型号之后，重新编译内核。

练习4——Kali live-build使用合适的工具完成工作

在 *Mr Robot* 影片的一个场景中，Angela 从 U 盘启动了一个 Kali，键入了几个命令来完成她的工作。请你定制一款 Kali ISO 镜像，来帮助 Angela 在启动 Kali 后无须触碰键盘，自动化完成她的工作。

视频链接：

`https://vimeo.com/225665986`

练习5——Kali live-build自动最小化安装

1．创建自安装的 ISO 镜像，该镜像要包含尽量少的软件包，仅包括 openssh-server 和 salt-minion。

2．为便于后续访问，请在该镜像内添加公钥。

3．要确保该 ISO 镜像正确完成，我们会在下一章使用它。

练习6——制作有多个持久化存储和LUKS核密码的Kali U盘

请制作一个 Kali 自生 U 盘，启用多个持久化存储，并启用 LUKS 核密码。

第10章　Kali Linux企业级应用

内容

- 通过网络安装 Kali Linux（PXE 启动）
- 高级配置管理
- 扩展和定制 Kali Linux
- 小结

目前，我们已经了解，Kali 是一款源自于 Debian 的强大的安全产品，它具备很高的安全性，提供高级软件包管理功能，支持多平台，且最负盛名的是，它是一个专业的安全工具武器库。那么，如何将其大规模部署，甚至应用到企业级呢。本章会详细介绍，Kali 远远不止可以在桌面级使用，它还提供了中心化管理和企业级控制能力。简而言之，通读本章之后，你将能够快速部署贴合自己需求的、安全性更高的 Kali 系统，以及通过 Kali 安装更新包，保持机器同步。

达到这种规模层次需要以下几步：包括启动 PXE 网络引导，使用高级配置管理工具（SaltStack），Fork 和定制化包，部署软件包。我们将详细介绍每一步，并展示如何摆脱繁重工作，轻松部署、管理和维护大量定制化的 Kali Linux 系统。如果这还不够，我们会投入一群助手，辅助管理你的帝国。

10.1　通过网络安装Kali Linux（PXE启动）

经过前面几章的学习，熟悉了以后，基础版 Kali Linux 的安装变得非常简单和直接。但如果需要同时安装多台机器，再使用这种安装方式，就相当枯燥乏味了。幸运的是，Kali 支持网络安装。这样的话，就可以同时在多台机器上简单快速地安装 Kali Linux 了。

首先，需要从网络启动目标机器。这一步可以借助于预启动执行环境（Preboot eXecution Environment，PXE），PXE 工作于 Client/Server 网络模式，设计用于从网络启动任意联网机器，且不论目标机器是否已安装操作系统。设置 PXE 网络启动，需要先配置一台 TFTP

（Trivial File Transfer Protocol）服务器，以及一台 DHCP/BOOTP 服务器。你可能还需要准备一台 Web 服务器，用于存放在安装过程中自动使用的 debconf 预生成文件。

幸运的是，dnsmasq 同时支持 DHCP 和 TFTP，因此仅需单个服务，即可满足所有需求。并且 Apache Web 服务器在 Kali 系统上也是默认安装的（但默认没启用）。

分开配置 DHCP 和 TFTP	在某些更复杂的安装场景中，dnsmasq 的功能可能会相当受限，或者你希望在一个已经运行 DHCP 服务的网络中，启用 PXE。以上两种情况，均需要分别配置 DHCP 和 TFTP 服务。 Debian 的安装手册，涵盖了针对 PXE 启动的 isc-dhcp-server 和 tftpd-hpa 服务的安装过程。 ➨ `https://www.debian.org/releases/stable/amd64/ch04s05.html`

需要通过`/etc/dnsmasq.conf` 文件来配置 dnsmasq。一个基础版的配置样例如下：

```
# Network interface to handle
interface=eth0
# DHCP options
# IP range to allocate
dhcp-range=192.168.101.100,192.168.101.200,12h
# Gateway to announce to clients
dhcp-option=option:router,192.168.101.1
# DNS servers to announce to clients
dhcp-option=option:dns-server,8.8.8.8,8.8.4.4
# Boot file to announce to clients
dhcp-boot=pxelinux.0
# TFTP options
enable-tftp
# Directory hosting files to serve
tftp-root=/tftpboot/
```

`/etc/dnsmasq.conf` 文件配置完毕后，需要将安装启动文件放到`/tftpboot/`目录下。Kali Linux 提供了专用压缩包文件，可直接下载并解压至`/tftpboot/`。从下列 URL，选择所需的压缩包文件：32 位（i386）还是 64 位（amd64），标准安装还是图形化（gtk）安装。

➨ `http://http.kali.org/dists/kali-rolling/main/installer-amd64/current/images/netboot/gtk/ netboot.tar.gz`

➨ `http://http.kali.org/dists/kali-rolling/main/installer-amd64/current/images/netboot/ netboot.tar.gz`

➥ http://http.kali.org/dists/kali-rolling/main/installer-i386/current/images/netboot/gtk/ netboot.tar.gz

➥ http://http.kali.org/dists/kali-rolling/main/installer-i386/current/images/netboot/netboot. tar.gz

选定压缩包后，创建/tftpboot/目录，下载压缩包，并将其解压至该目录。

```
# mkdir /tftpboot
# cd /tftpboot
# wget http://http.kali.org/dists/kali-rolling/main/installer-
amd64/current/images/
   ➥ netboot/netboot.tar.gz # tar xf netboot.tar.gz
# tar xf netboot.tar.gz
# ls -l
total 25896
drwxrwxr-x 3 root root    4096 May  6 04:43 debian-installer
lrwxrwxrwx 1 root root 47 May 6 04:43 ldlinux.c32 -> debian-
installer/amd64/boot
   ➥ -screens/ldlinux.c32
-rw-r--r-- 1 root root 26507247 May 6 04:43 netboot.tar.gz
lrwxrwxrwx 1 root root 33 May 6 04:43 pxelinux.0 -> debian-installer/amd64/
   ➥ pxelinux.0
lrwxrwxrwx 1 root root 35 May 6 04:43 pxelinux.cfg -> debian-
installer/amd64/
   ➥ pxelinux.cfg
-rw-rw-r-- 1 root root 71 May 6 04:43 version.info
```

解压的文件包括 pxelinux 启动引导程序，该程序与 syslinux 和 isolinux 使用相同的配置文件。因此，在定制化 Kali Linux 的自生（Live）ISO 镜像时，就可根据需要随意调整 debian-installer/amd64/boot-screens/目录下的启动文件。

举例来说，假设你选择了文本安装，你可以通过添加启动参数来预设语言、国家、键盘布局、主机名以及域名等值。还可以引用外部预设文件的 URL，以及配置超时时间（例如 5s 内没有按键响应），自动执行启动程序等。这些动作，都可以通过修改 debian-installer/amd64/txt.cfg 配置文件来实现：

```
label install
       menu label ^Install
       kernel debian-installer/amd64/linux
       append vga=788 initrd=debian-installer/amd64/initrd.gz --- quiet
          ➥ language=en country=US keymap=us hostname=kali domain=
          ➥ url=http://192.168.101.1/preseed.cfg
```

然后需要修改 debian-installer/amd64/syslinux.cfg 配置文件，来调整超时时间：

```
# D-I config version 2.0
# search path for the c32 support libraries (libcom32, libutil etc.)
path debian-installer/amd64/boot-screens/
include debian-installer/amd64/boot-screens/menu.cfg
default debian-installer/amd64/boot-screens/vesamenu.c32
prompt 0
timeout 50
```

具备从网络 PXE 启动任意机器的能力后，即可利用 4.3 节所述的方法，在多台机器上，无须物理启动介质，实现完全启动、预设参数、静默安装等功能。当然，别忘了灵活运用启动参数 url=http://server/preseed.cfg，它是一个基于网络的预置文件。

10.2 高级配置管理

具备同时快速安装多台 Kali 的能力之后，你可能需要在这些机器安装完毕后，进行配置管理。此时，可使用配置管理工具来管理机器。

Kali Linux 有很多大受欢迎的配置管理工具，比如 ansible、chef、puppet、saltstack 等。但在本章，我们只讲 SaltStack。

➨ https://saltstack.com

10.2.1 配置SaltStack

SaltStack 是一款中心化的配置管理服务：一个 salt master 管理多个 salt minion。先准备一台服务器，安装 salt-master 软件包，在其他需要管理的主机上，安装 salt-minion，确保服务器与主机之间的连接正常。每个 salt minion 需要告知如何找到 salt master，编辑 /etc/salt/minion 文件，设置 master 值为 salt master 的域名或 IP 地址。请注意，*SaltStack* 使用 YAML 格式的配置文件。

```
minion# vim /etc/salt/minion
minion# grep ^master /etc/salt/minion
master: 192.168.122.105
```

每个 *salt minion* 对应一个独有的标识符，存储在/etc/salt/minion_id 文件内，默认

值是其主机名。这个标识符，会被用于后续的配置过程中，因此它非常重要，理应在 minion 与 master 通信之前设置好。

```
minion# echo kali-scratch >/etc/salt/minion_id
minion# systemctl enable salt-minion
minion# systemctl start salt-minion
```

salt-minion 服务正常启动之后，它会尝试与 salt master 通信，并交换一些密钥。在 master 一端，需主动接受 minion 用于标识自身的密钥，以便于后续的通信能继续下去：

```
master# systemctl enable salt-master
master# systemctl start salt-master
master# salt-key --list all Accepted Keys:
Denied Keys:
Unaccepted Keys: kali-scratch
Rejected Keys:
master# salt-key --accept kali-scratch
The following keys are going to be accepted:
Unaccepted Keys:
kali-scratch
Proceed? [n/Y] y
Key for minion kali-scratch accepted.
```

10.2.2　在minion上执行命令

minion 连通之后，即可从 master 上执行命令：

```
master# salt '*' test.ping
kali-scratch:
    True
kali-master:
    True
```

该命令要求所有 minion（*是通配符，表示所有的 minion）执行 test 模块的 ping 功能。该功能成功执行会返回 True，这是测试 master 和 minion 之间正常通信最简单的方式之一。

指定 minion 执行命令的方法是：在第一个参数中，指定一个 minion 的身份标识符，或者字母加*（如*-sratch 或 kali-*）的方式，指定所有 minion 的一个子集。以下是一个例子，它描述了如何在 kali-scratch 这个 minion 上，执行任意 shell 命令。

```
master# salt kali-scratch cmd.shell 'uptime; uname -a'
```

```
kali-scratch:
    05:25:48 up 44 min, 2 users, load average: 0.00, 0.01, 0.05
    Linux kali-scratch 4.5.0-kali1-amd64 #1 SMP Debian 4.5.3-2kali1
        ➥ (2016-05-09) x86_64 GNU/Linux
```

Salt 模块介绍

因应用场景不同，执行模块也非常多，这里不打算介绍所有模块。想学习了解所有模块，可前往这个链接：https://docs.saltstack.com/en/latest/ref/modules/all/index.html。还可以通过执行 salt minion sys.doc 命令，查看所有执行模块及其功能的描述文档。该命令执行的返回结果相当长。所以可以通过附加父模块作为前缀，功能或模块名称作为参数，来对结果进行过滤。

```
master# salt kali-scratch sys.doc disk.usage
disk.usage:

    Return usage information for volumes mounted on this
      ➥ minion
```

还有个很实用的模块是 pkg 模块，它是一个包管理器的抽象，依赖于当前操作系统的包管理器（比如 Debian 及其衍生系统 Kali 的 apt-get）。pkg.refresh_db 命令，可更新包的缓存列表（执行 apt-get update 命令）；pkg.upgrade 命令，可安装所有可用的更新包（执行 apt-get upgrade 或 apt-get dist-upgrade 命令，具体哪个命令，取决于执行参数）；pkg.list_upgrades 命令，可列出所有待更新操作（同 pkg.upgrade dist_upgrade=True 命令的执行效果一样）。

service 模块，是服务管理器的抽象（如 Kali 系统的 systemd），它允许执行所有常见 systemctl 操作：service.enable、service.disable、service.start、service.stop、service.restart 以及 service.reload：

```
master# salt '*' service.enable ssh
kali-scratch:
    True
kali-master:
    True
master# salt '*' service.start ssh
kali-master:
    True
kali-scratch:
    True
master# salt '*' pkg.refresh_db
kali-scratch:
```

```
    ----------
kali-master:
    ----------
master# salt '*' pkg.upgrade dist_upgrade=True
kali-scratch:
    ----------
    changes:
        ----------
        base-files:
            ----------
            new:
                1:2016.2.1
            old:
                1:2016.2.0
[...]
        zaproxy:
            ----------
            new:
                2.5.0-0kali1
            old:
                2.4.3-0kali3
    comment:
    result:
        True
```

举一个更具体的例子，可使用 `salt` 非常便捷地安装 Nmap 分布式版本 dnmap。在所有 minion 上均安装完毕后，即可在终端上启动服务。

```
server# salt '*' pkg.install dnmap
[...]
server# vim dnmap.txt
server# dnmap_server -f dnmap.txt
```

假设服务器 IP 为 1.2.3.4，然后即可告诉所有 minion 启动一个客户端进程，连接服务器。

```
server# salt '*' cmd.run_bg template=jinja 'dnmap_client -s 1.2.3.4 -a \
    ➥ {{ grains.id }}'
kali-scratch:
    ----------
    pid:
        17137
[...]
```

注意，在本例中，我们使用 `cmd.run_bg`，在后台运行了 `dnmap_client` 命令。不要

等它执行结束，因为它执行耗时很长。而且不幸的是，如果进程被中断，它无法正确关闭自己，所以你还需要做清理工作：

```
server# salt '*' cmd.shell 'pkill -f dnmap_client'
```

10.2.3 Salt State及其他功能

虽然远程执行是 SaltStack 非常重要的功能，但这实际上仅仅是 SaltStack 功能的一小部分而已。

在配置新机器时，经常需要运行多个命令和测试脚本，以确定系统安装之前的详细信息。这些操作，可以归整为可重用的配置模板，这些配置模板就叫作 state 文件。

state 文件中描述的操作，会分别由 state.apply 命令执行。

为节约时间，还可以依赖社区创建的众多现有 state 文件。这些文件可在“Salt formulas”里找到：

➨ https://docs.saltstack.com/en/latest/topics/development/conventions/formulas.html

可组合起来的功能有很多：

- 执行动作调度
- 定义 minion 触发事件的响应动作
- 从 minion 收集数据
- 涉及多个 minion 的一系列操作的编排工作
- 未安装 salt-minion 服务时，通过 SSH 应用状态
- 在云架构中管理它们
- 其他

SaltStack 非常庞大，这里无法介绍它的所有功能。有一些专门讲解 SaltStack 的书籍，其在线文档也非常丰富。如果想要深入了解，请前往以下链接：

➨ https://docs.saltstack.com/en/latest/

如果你正在管理大量机器，建议多了解一些 SaltStack，它不仅可以在部署新机器时节省大量时间，还可以在整个网络中保持配置的一致性。

为了对 state 文件有个直观的理解，这里通过一个简单的例子说明，如何启用 apt 源以及如何安装软件包，如 10.3.3 节和 10.3.2 节所述。你还需要提前为 root 账号注册 SSH 公钥，以便在发生问题时，可以远程登录解决。

默认情况下，state 文件存储在 master 节点的/srv/salt 目录下；文件为 YAML 格式，其扩展名为.sls。应用一个 state，跟运行命令一样，依赖许多其他的 state 模块。

➡ https://docs.saltstack.com/en/latest/topics/tutorials/starting_states.html

➡ https://docs.saltstack.com/en/latest/ref/states/all/

在该例中，/srv/salt/offsec.sls 文件需要调用三个模块：

```
offsec_repository:
   pkgrepo.managed:
     - name: deb http://pkgrepo.offsec.com offsec-internal main
     - file: /etc/apt/sources.list.d/offsec.list
     - key_url: salt://offsec-apt-key.asc
     - require_in:
        - pkg: offsec-defaults

offsec-defaults:
  pkg.installed

ssh_key_for_root:
 ssh_auth.present:
    - user: root
    - name: ssh-rsa AAAAB3NzaC1yc2...89C4N rhertzog@kali
```

offsec_repository 依赖 pkgrepo 模块。在该例中，使用了 pkgrepo 模块的 managed 方法，注册了一个软件源。key_url 属性是一个 ASCII 编码的 GPG 密钥，用于验证软件源的签名，该密钥可从 salt 的 Master 节点/srv/salt/offsec-apt-key.asc 获取。require_in 属性表示，这个状态需要在 offsec-defaults 状态之前执行，因为 offsec-defaults 状态要求正确配置好软件源，才可以安装相应的包。

offsec-defaults 状态安装了同名的软件包。由此可见，state 的名称非常重要，虽然也可以被 name 属性覆盖。如同该例中 state 的名称，较为简洁，同时具有较好的可读性。

最后一个状态 ssh_key_for_root，通过 name 属性，向/root/.ssh/authorized_keys 文件内，添加了一个 SSH key。注意，这里为了可读性省略了部分内容，自己在填写时，要在 name 属性里写全。

然后，即可以在指定的 minion 里应用这个 state 文件。

```
server# salt kali-scratch state.apply offsec
kali-scratch:
----------
```

```
        ID: offsec_repository
   Function: pkgrepo.managed
      Name: deb http://pkgrepo.offsec.com offsec-internal main Result: True
    Comment: Configured package repo 'deb http://pkgrepo.offsec.com offsec-
       ➥ internal main'
    Started: 06:00:15.767794
   Duration: 4707.35 ms
    Changes:
             ----------
             repo:
                  deb http://pkgrepo.offsec.com offsec-internal main
----------
        ID: offsec-defaults
   Function: pkg.installed
     Result: True
    Comment: The following packages were installed/updated: offsec-defaults
    Started: 06:00:21.325184
   Duration: 19246.041 ms
    Changes:
             ----------
             offsec-defaults:
                 ----------
                 new:
                   1.0
                 old:
----------
       ID: ssh_key_for_root
  Function: ssh_auth.present
      Name: ssh-rsa AAAAB3NzaC1yc2...89C4N rhertzog@kali
    Result: True
   Comment: The authorized host key AAAAB3NzaC1yc2...89C4N for user root
was added
   Started: 06:00:40.582539
  Duration: 62.103 ms
   Changes:
            ----------
            AAAAB3NzaC1yc2...89C4N:
              New

Summary for kali-scratch
------------
Succeeded: 3 (changed=3)
Failed:    0
```

```
------------
Total states run:    3
Total run time:  24.015 s
```

可通过在/srv/salt/top.sls 文件中记录，永久性地将其关联到指定 minion 中，然后，即可使用 state.highstate 命令，一键应用所有相关的 state。

```
server# cat /srv/salt/top.sls
base:
  kali-scratch:
    - offsec
server# salt kali-scratch state.highstate
kali-scratch:
----------
         ID: offsec_repository
    Function: pkgrepo.managed
       Name: deb http://pkgrepo.offsec.com offsec-internal main Result: True
     Comment: Package repo 'deb http://pkgrepo.offsec.com offsec-internal
       ➥ main' already configured
     Started: 06:06:20.650053
    Duration: 62.805 ms
     Changes:
----------
         ID: offsec-defaults
    Function: pkg.installed
      Result: True
     Comment: Package offsec-defaults is already installed
     Started: 06:06:21.436193
    Duration: 385.092 ms
     Changes:
----------
         ID: ssh_key_for_root
    Function: ssh_auth.present
       Name: ssh-rsa AAAAB3NzaC1yc2...89C4N rhertzog@kali
      Result: True
     Comment: The authorized host key AAAAB3NzaC1yc2...89C4N is already
       ➥ present for user root
     Started: 06:06:21.821811
    Duration: 1.936 ms
     Changes:

Summary for kali-scratch
------------
```

```
Succeeded: 3
Failed:   0
------------
Total states run:    3
Total run time: 449.833 ms
```

10.3 扩展和定制Kali Linux

为满足自身特定需求，常常需要对 Kali Linux 进行改动。最好的方式是，维护属于自己的代码仓库，将原 Kali 软件包 fork 进来，将修改后版本托管于这个代码仓库中，这同样适用于配置定制化的增补包和非 Kali Linux 官方的外部包。

10.3.1 fork Kali软件包

关于这个主题，请参考 9.1 节中的介绍。

只要你愿意，所有的软件包均可以 fork。但是需要注意的是，fork 软件包是需要一定开销的，因为每当 Kali 发布更新后，你都需要去更新它。以下是几个不得不 fork 的理由：

- 给某个 bug 打补丁，或者添加新功能。大部分时候，你更想把这个补丁提交给上游开发者，以从源头修复这个 bug 或添加新功能。
- 选择其他编译选项重编译（假设有充分的理由表明，为何 Kali 不使用这些编译选项；否则，最好与 Kali 的开发者们讨论，看看他们是否可以启用这些编译选项）。

相比之下，以下几种情况则不建议 fork 软件包，如果不 fork，这里也会给出意见来帮助处理这些问题。

- 修改配置文件。在这种情况下，有多种更好的选择。比如使用配置管理器，来自动化安装修改后的配置文件；或者安装配置软件包，它会将文件放入配置目录，或者转移原始配置文件。
- 更新一个上游版本。再强调一遍，在这种情况下，最好直接与 Debian 或 Kali 的开发者直接沟通协作完成更新工作。如果是滚动发行版，更新包会很快抵达终端用户。

在现有软件包中，有一些是 Kali Linux 的构件，在某些情况下，fork 它们还是挺有意思的。

- kali-meta：它用于编译所有 kali-linux-*软件包，尤其是 kali-linux-full 软件包，它定义了 Kali Linux ISO 镜像内默认应安装哪些软件包。
- desktop-base：它包含桌面安装中涉及的各类杂项文件。如果你想 show 一下自己的

logo 或者换个桌面主题，那么可以考虑 fork 这个源码包。

- kali-menu：它定义了 Kali 的菜单结构，同时有个`.desktop` 文件，用于存放所有 Kali 菜单中列出的应用程序。

10.3.2　新建配置包

到目前为止，我们已经接触了 PXE 启动，也介绍了如何通过 SaltStack 进行配置管理，还讨论了软件包 fork。接下来，我们将这些过程打包成一个具体例子，并通过创建配置定制软件包，在多台机器上半自动地部署定制的配置文件。

在这个例子中，我们将新定制一个软件包，它可以使用你自己的软件包仓库和 GnuPG 签名密钥，分发 SaltStack 配置，推送定制的桌面背景，以及统一向所有 Kali 系统提供默认的桌面环境。

这似乎是一项艰巨的任务（尤其是看过Debian New Maintainer Guide[1]之后）。幸运的是，配置软件包，主要就是个复杂文件的打包，把它转成一个软件包，相对比较容易。

分析一个软件包样例　假如你想深入分析一个实际的配置包，可以参考 kali-defaults 软件包。它不像本节其他示例那样简单，它具有所有相关特性，甚至使用了一些高级技巧（比如`dpkg-divert`），来替换那些已由其他软件包提供的文件。

offsec-defaults 包含以下文件。

- `/etc/apt/sources.list.d/offsec.list`：apt 工具的源列表，启用了公司的内部软件源。
- `/etc/apt/trusted.gpg.d/offsec.gpg`：GnuPG 签名密钥，用于公司内部软件包的签名。
- `/etc/salt/minion.d/offsec.conf`：SaltStack 配置文件，指明如何找到 Salt master。
- `/usr/share/images/offsec/background.png`：一张有 Offensive Security 公司 logo 的背景图片。
- `/usr/share/glib-2.0/schemas/90_offsec-defaults.gschema.override`：GNOME 桌面配置文件。

首先，新建 `offsec-defaults-1.0` 目录，放入所有文件。然后运行 `dh_make --`

1 https://www.debian.org/doc/manuals/maint-guide/

native 命令（dh-make 软件包），添加 Debian 打包指令。所有生成文件会存放在 debian 子目录下。

```
$ mkdir offsec-defaults-1.0; cd offsec-defaults-1.0
$ dh_make --native
Type of package: (single, indep, library, python)
[s/i/l/p]? i
Email-Address      : buxy@kali.org
License            : gpl3
Package Name       : offsec-defaults
Maintainer Name    : Raphaël Hertzog
Version            : 1.0
Package Type       : indep
Date               : Thu, 16 Jun 2016 18:04:21 +0200
Are the details correct? [Y/n/q] y
Currently there is not top level Makefile. This may require additional
tuning
Done. Please edit the files in the debian/ subdirectory now.
```

首先，提示你选择输入软件包类型。这里选择 i（indep），它表示这个源码包会生成单个二进制包，该包可以在所有处理器架构之间共享（Architecture: all）。single 与之类似，但它生成的二进制包，依赖目标处理器架构（Architecture: any）。这样的话，indep 选项更适合些，因为源码包只包含文本文件，没有二进制程序，所以可以用在所有架构之上。library 更适用于共享库，因为它们需要严格遵循打包规则。与之类似，python 选项仅用于 Python 模块。

项目维护者的姓名和邮箱

大多数包管理维护程序，都会从 DEBFULLNAME 和 DEBEMAIL/EMAIL 环境变量去找项目维护者的姓名和邮箱。定义一次，终身受用，可以免去重复码字的痛苦。如果你的 shell 是 Bash，简单的做法是在~/.bashrc 里添加如下两行：

```
export EMAIL="buxy@kali.org"
export DEBFULLNAME="Raphael Hertzog"
```

dh_make 命令创建了包含很多文件的 debian 子目录。有些是必需的，尤其是 rules、control、changelog 和 copyright 文件。以.ex 扩展名结尾的文件是一些示例文件，可以直接修改其中的内容，最后移除.ex 扩展名。如果不需要它们，建议把这些文件删除。compat 文件不能删除，因为它定义了 debhelper 兼容级别，debhelper 套件在包构建的各个阶段均会用到。

copyright 文件必须包含作者信息以及相关许可（license）。如果 dh_make 默认选择

的 license 不适合你，你就需要编辑这个文件。以下是一个 copyright 文件的改动版本：

```
Format: https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/
Upstream-Name: offsec-defaults

Files: *
Copyright: 2016 Offensive Security
License: GPL-3.0+

License: GPL-3.0+
 This program is free software: you can redistribute it and/or modify
 it under the terms of the GNU General Public License as published by
 the Free Software Foundation, either version 3 of the License, or
 (at your option) any later version.
 .
 This package is distributed in the hope that it will be useful,
 but WITHOUT ANY WARRANTY; without even the implied warranty of
 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 GNU General Public License for more details.
 .
 You should have received a copy of the GNU General Public License
 along with this program. If not, see <https://www.gnu.org/licenses/>.
 .
 On Debian systems, the complete text of the GNU General
 Public License version 3 can be found in "/usr/share/common-licenses/GPL-3".
```

changelog 文件都比较类似，把“Initial release”改为信息量更大的一些字符描述，就差不多了。

```
offsec-defaults (1.0) unstable; urgency=medium
  * Add salt minion’s configuration file.
  * Add an APT’s sources.list entry and an APT's trusted GPG key.
  * Override the gsettings schema defining the background picture.

  -- Raphaël Hertzog <buxy@kali.org> Thu, 16 Jun 2016 18:04:21 +0200
```

在这个例子中，我们对 control 文件做了些改动。把 Section 的值改为 misc，移动了 Homepage、Vcs-Git、Vcs-Browser 字段。最后，在 Description 字段内填写了具体信息：

```
Source: offsec-defaults
Section: misc
Priority: optional
```

```
Maintainer: Raphaël Hertzog <buxy@kali.org>
Build-Depends: debhelper (>= 9)
Standards-Version: 3.9.8

Package: offsec-defaults
Architecture: all
Depends: ${misc:Depends}
Description: Default settings for Offensive Security
 This package contains multiple files to configure computers
 owned by Offensive Security.
 .
 It notably modifies:
  - APT's configuration
  - salt-minion's configuration
  - the default desktop settings
```

`rules` 文件包含若干 rule，这些 rule 用于编译构建以及在指定子目录（以生成的二进制包命名）内安装软件等。该子目录的内容会被归档至 Debian 软件包中。在这种情况下，文件会被安装在 `debian/offsec-defaults/` 子目录下。例如，某个包最终安装了 `/etc/apt/sources.list. d/offsec.list` 文件，该文件会在 `debian/offsec-defaults/etc/apt/sources.list.d/offsec.list` 下。`rules` 文件会被当作 `Makefile` 文件来用，部分 target 是标准通用的（包括 `clean` 和 `binary`，分别用于清理源码目录和生成二进制包）。

什么是 Makefile 文件

你可能已经注意到了在 `dh_make` 输出的最后部分有关缺少 Makefile 文件的消息，也注意到它与 `rules` 文件的相似性。`Makefile` 文件就是一个脚本文件，它用于编译构建程序。它描述了一些规则，如何在一个树状依赖环境中，编译构建一组文件。比如，一个程序可以从一组源码文件中编译构建。`Makefile` 文件使用如下格式描述这些规则：

```
target: source1 source2 ...
        command1
        command2
```

这条 rule 的解释如下：如果某个 `source*`文件，时间上比 `target` 文件新，则 `target` 文件需使用 `command1` 和 `command2` 命令重新生成。

注意，命令所在行须以 tab 开头；同样需要注意的是，如果命令行以短破折号（`-`）开头，则命令执行失败时，不会打断整个构建过程。

虽然该文件是核心文件，但它只包含了用于运行 debhelper 标准命令集的最低限度集合，也就是 dh_make 所生成的文件。多数情况下，建议创建 debian/offsec-defaults.install 文件，来配置 dh_install 命令的行为：

```
apt/offsec.list etc/apt/sources.list.d/
apt/offsec.gpg etc/apt/trusted.gpg.d/
salt/offsec.conf etc/salt/minion.d/
images/background.png usr/share/images/offsec/
```

还可以用它来安装 gsetting 覆盖文件，但 debhelper 提供了一个专用于此的工具（dh_installgsettings）。要使用它，把你的设置放到 debian/offsec-defaults.gsettings-override 文件内即可：

```
[org.gnome.desktop.background]
picture-options='zoom'
picture-uri='file:///usr/share/images/offsec/background.png'
```

接下来，重写 debian/rules 文件内的 dh_installgsettings 调用，增加其优先级至组织级别（根据手册，其值为 90）。

```
#!/usr/bin/make -f

%:
	dh $@

override_dh_installgsettings:
	 dh_installgsettings --priority=90
```

此时，源码包已准备完毕。剩下的工作，就是生成二进制包，使用前面重编译构建包同样的方法：在 offsec-defaults-1.0 目录下，运行 dpkg-buildpackage -us -uc 命令。

```
$ dpkg-buildpackage -us -uc
dpkg-buildpackage: info: source package offsec-defaults
dpkg-buildpackage: info: source version 1.0
dpkg-buildpackage: info: source distribution unstable
dpkg-buildpackage: info: source changed by Raphaël Hertzog <buxy@kali.org>
dpkg-buildpackage: info: host architecture amd64
dpkg-source --before-build offsec-defaults-1.0
 fakeroot debian/rules clean
dh clean
   dh_testdir
```

```
   dh_auto_clean
   dh_clean
dpkg-source -b offsec-defaults-1.0
dpkg-source: info: using source format ’3.0 (native)’
dpkg-source: info: building offsec-defaults in offsec-defaults_1.0.tar.xz
dpkg-source: info: building offsec-defaults in offsec-defaults_1.0.dsc
 debian/rules build
dh build
   dh_testdir
   dh_update_autotools_config
   dh_auto_configure
   dh_auto_build
   dh_auto_test
 fakeroot debian/rules binary
dh binary
   dh_testroot
   dh_prep
   dh_auto_install
   dh_install
   dh_installdocs
   dh_installchangelogs
   debian/rules override_dh_installgsettings
make[1]: Entering directory ’/home/rhertzog/kali/kali-book/samples/offsec-
defaults-1.0’
dh_installgsettings --priority=90
make[1]: Leaving directory ’/home/rhertzog/kali/kali-book/samples/offsec-
defaults-1.0’
   dh_perl
   dh_link
   dh_strip_nondeterminism
   dh_compress
   dh_fixperms
   dh_installdeb
   dh_gencontrol
   dh_md5sums
   dh_builddeb
dpkg-deb: building package ’offsec-defaults’ in ’../offsec-
defaults_1.0_all.deb’.
 dpkg-genchanges >../offsec-defaults_1.0_amd64.changes
dpkg-genchanges: info: including full source code in upload
 dpkg-source --after-build offsec-defaults-1.0
dpkg-buildpackage: info: full upload; Debian-native package (full source is
included)
```

10.3.3　为APT创建一个软件包仓库

现在你已经有了一个定制包，你可以通过一个 APT 软件源进行分发。使用 `reprepro` 创建所需的仓库，并填充之。该工具相当强大，其帮助手册绝对值得一读。

软件包仓库通常托管在服务器上。为正确地将其与服务器上其他服务区分开来，最好创建一个该服务的专用用户。在专用用户中，可以托管仓库文件和用于包签名的 GnuPG 密钥。

```
# apt install reprepro gnupg
[...]
# adduser --system --group pkgrepo
Adding system user ‘pkgrepo’ (UID 136) ...
Adding new group ‘pkgrepo’ (GID 142) ...
Adding new user ‘pkgrepo’ (UID 136) with group ‘pkgrepo’ ...
Creating home directory ‘/home/pkgrepo’ ...
# chown pkgrepo $(tty)
# su - -s /bin/bash pkgrepo
$ gpg --gen-key
gpg (GnuPG) 2.1.11; Copyright (C) 2016 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

gpg: directory ’/home/pkgrepo/.gnupg’ created
gpg: new configuration file ’/home/pkgrepo/.gnupg/dirmngr.conf’ created
gpg: new configuration file ’/home/pkgrepo/.gnupg/gpg.conf’ created
gpg: keybox ’/home/pkgrepo/.gnupg/pubring.kbx’ created
Note: Use ”gpg --full-gen-key” for a full featured key generation dialog.

GnuPG needs to construct a user ID to identify your key.

Real name: Offensive Security Repository Signing Key
Email address: repoadmin@offsec.com
You selected this USER-ID:
   ”Offensive Security Repository Signing Key <repoadmin@offsec.com>”
Change (N)ame, (E)mail, or (O)kay/(Q)uit? o
We need to generate a lot of random bytes. It is a good idea to perform some
other action (type on the keyboard, move the mouse, utilize the disks)
during the prime generation; this gives the random number generator a better
chance to gain enough entropy.
[...]
gpg: /home/pkgrepo/.gnupg/trustdb.gpg: trustdb created
gpg: key B4EF2D0D marked as ultimately trusted
```

```
gpg: directory '/home/pkgrepo/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as '/home/pkgrepo/.gnupg/openpgp-
    ➥ revocs.d/F8FE22F74F1B714E38DA6181B27F74F7B4EF2D0D.rev'
public and secret key created and signed.

gpg: checking the trustdb
gpg: marginals needed: 3 completes needed: 1 trust model: PGP
gpg: depth: 0 valid: 1 signed: 0 trust: 0-, 0q, 0n, 0m, 0f, 1u
pub rsa2048/B4EF2D0D 2016-06-17 [S]
    Key fingerprint = F8FE 22F7 4F1B 714E 38DA 6181 B27F 74F7 B4EF 2D0D
uid       [ultimate] Offensive Security Repository Signing Key
<repoadmin@offsec.com>
sub rsa2048/38035F38 2016-06-17 []
```

注意，如果希望在签名仓库时无须人工交互，当提示输入密码时，应输入一个空值（并确认不想保护你的私钥）。还要注意，`gpg` 要求拥有虚拟终端的写权限，以便于能够安全地提示输入密码。这就是为何在以 pkgrepo 用户身份启动 shell 之前，你需要先更改虚拟终端的所有者的原因（虚拟终端是属于 root 的，最初是以 root 用户启动它）。

现在可以开始设置仓库了。`reprepro` 需要一个专用目录，且在该目录内，你已创建了 `conf/distributions` 文件，该文件说明你在软件包仓库中有哪些可用版本。

```
$ mkdir -p reprepro/conf
$ cd reprepro
$ cat >conf/distributions <<END
Codename: offsec-internal
AlsoAcceptFor: unstable
Origin: Offensive Security
Description: Offsec's Internal packages
Architectures: source amd64 i386
Components: main
SignWith: F8FE22F74F1B714E38DA6181B27F74F7B4EF2D0D
END
```

必须填写的字段是 `Codename`，它用于记录版本名称；`Architectures` 字段，用于说明该版本可适用于哪些体系架构；`Components` 字段，说明该版本有哪些可用组件（组件是版本的细分，可在 APT 的源列表 `sources.list` 中开启）；`Origin` 和 `Description` 字段，是纯信息字段，从 Release 文件中直接复制它们；`SignWith` 字段，要求 `reprepro` 使用 GnuPG 密钥对仓库进行签名（这里给出了完整的指纹信息，是为了确保使用正确的密钥签名，同时避免短标识符冲突）；`AlsoAcceptFor` 字段，不是必需的，它可用于处理`.changes` 文件，该文件的 `Distribution` 字段，在这里有一个值与之对应（如果没有，

它只接受该字段中版本的 codename）。

有了这些基本设置，即可使用 reprepro 生成一个空的仓库：

```
$ reprepro export
Exporting indices...
$ find .
.
./db
./db/version
./db/references.db
./db/contents.cache.db
./db/checksums.db
./db/packages.db
./db/release.caches.db
./conf
./conf/distributions
./dists
./dists/offsec-internal
./dists/offsec-internal/Release.gpg
./dists/offsec-internal/Release
./dists/offsec-internal/main
./dists/offsec-internal/main/source
./dists/offsec-internal/main/source/Release
./dists/offsec-internal/main/source/Sources.gz
./dists/offsec-internal/main/binary-amd64
./dists/offsec-internal/main/binary-amd64/Packages
./dists/offsec-internal/main/binary-amd64/Release
./dists/offsec-internal/main/binary-amd64/Packages.gz
./dists/offsec-internal/main/binary-i386
./dists/offsec-internal/main/binary-i386/Packages
./dists/offsec-internal/main/binary-i386/Release
./dists/offsec-internal/main/binary-i386/Packages.gz
./dists/offsec-internal/InRelease
```

如你所见，reprepro 在 dists 子目录下，创建了仓库元信息。同样，在 db 子目录下初始化了整个内部数据库。

现在可以添加你的软件包了。首先，复制 offsec-defaults 生成的文件（offsec-defaults_1.0.dsc 、 offsec-defaults_1.0.tar.xz 、 offsec-defaults_1.0_all.deb 和 offsec-defaults_1.0_amd64.changes）到服务器的/tmp 目录下，然后使用 reprepro 命令包含这些文件。

```
$ reprepro include offsec-internal /tmp/offsec-defaults_1.0_amd64.changes
Exporting indices...
$ find pool
pool
pool/main
pool/main/o pool/main/o/offsec-defaults
pool/main/o/offsec-defaults/offsec-defaults_1.0.dsc
pool/main/o/offsec-defaults/offsec-defaults_1.0.tar.xz
pool/main/o/offsec-defaults/offsec-defaults_1.0_all.deb
```

如上所示，它将这些文件添加到自己的位于 `pool` 子目录下的软件包池中。

对于 `dists` 和 `pool` 这两个目录，需要设置它们可以通过 HTTP 被公开访问来完成 APT 源的配置。这两个目录包括了所有 APT 需要下载的文件。

如果你想将其 host 到名为 pkgrepo.offsec.com 的虚拟主机上，则可以创建下面的 Apache 配置文件，并保存到`/etc/apache2/sites-available/pkgrepo.offsec.com.conf` 上，且使用 `a2ensite pkgrepo.offsec.com` 启动它。

```
<VirtualHost *:80>
    ServerName pkgrepo.offsec.com
    ServerAdmin repoadmin@offsec.com

    ErrorLog /var/log/apache2/pkgrepo.offsec.com-error.log
    CustomLog /var/log/apache2/pkgrepo.offsec.com-access.log "%h %l %u %t
    \"%r\" %>s %O"

    DocumentRoot /home/pkgrepo/reprepro

    <Directory "/home/pkgrepo/reprepro">
       Options Indexes FollowSymLinks MultiViews
       Require all granted
       AllowOverride All
    </Directory>
</VirtualHost>
```

对应需要添加的 `sources.list` 内的源列表，应如下所示：

```
deb http://pkgrepo.offsec.com offsec-internal main

# Enable next line if you want access to source packages too
# deb-src http://pkgrepo.offsec.com offsec-internal main
```

现在，你的软件包已经发布了，同时，应对你的所有网络主机开放访问。

虽然设置过程很烦琐，但总算完成了。你可以通过 PXE 启动联网机器，无须交互安装定制的 Kali Linux，通过 SaltStack 管理配置文件（以及控制 minion），创建 fork 的定制软件包，并通过自己的软件包仓库，分发这些软件包。这提供了针对多台 Kali Linux 的企业级控制和集中化管理。总之，你现在可以快速、高效、安全地部署 Kali 系统，并且可针对自己特定需求预先配置，保持所有 Kali 软件包同步更新。

10.4　小结

Kali Linux 可从桌面级部署扩展至中大规模部署，甚至到企业级部署。在本章，我们讨论了如何通过 SaltStack 集中化管理多个 Kali 系统，从而满足快速、安全、定制化的部署需求。还展示了如何通过 Kali 的更新包功能来保持系统同步。

还讨论了软件包 fork，这允许你创建定制的、可分发的源码包。

总之，我们来回顾一下建立 Salt Master 和 Minion 所需的主要步骤，这些步骤允许远程控制和配置远程主机。

要点提示：

- 使用 PXE 从网络启动的机器，至少需要一个 TFTP 文件服务器、一个 DHCP/BOOTP 服务器（以及一个用作 `debconf` 的 Web 服务器）。dnsmasq 可同时用作 DHCP 和 TFTP 服务器，Apache2 Web 服务器在 Kali 上是预装的（但默认没启用）。
- Debian 安装手册里有 PXE 启动所需的 isc-dhcp-server 和 tftpd-hpa 安装配置。

  ➨ `https://www.debian.org/releases/stable/amd64/ch04s05.html`
- dnsmasq 的配置文件为`/etc/dnsmasq.conf`，基础配置仅需以下几行：

```
# Network interface to handle
interface=eth0
# DHCP options
# IP range to allocate
dhcp-range=192.168.101.100,192.168.101.200,12h
# Gateway to announce to clients
dhcp-option=option:router,192.168.101.1
# DNS servers to announce to clients
dhcp-option=option:dns-server,8.8.8.8,8.8.4.4
# Boot file to announce to clients
dhcp-boot=pxelinux.0
# TFTP options
enable-tftp
# Directory hosting files to serve
```

```
tftp-root=/tftpboot/
```

- 解压网络启动文件的安装包至/tftpboot/目录，有 32 位（i386）、64 位（amd64）、标准版、图形版（gtk）之分。关于压缩包请参见以下链接：

➡ http://http.kali.org/dists/kali-rolling/main/installer-amd64/current/images/netboot/gtk/ netboot.tar.gz

➡ http://http.kali.org/dists/kali-rolling/main/installer-amd64/current/images/netboot/netboot. tar.gz

➡ http://http.kali.org/dists/kali-rolling/main/installer-i386/current/images/netboot/gtk/netboot. tar.gz

➡ http://http.kali.org/dists/kali-rolling/main/installer-i386/current/images/netboot/netboot.tar. gz

操作命令如下：

```
# mkdir /tftpboot
# cd /tftpboot
# wget http://http.kali.org/dists/kali-rolling/main/installer-
   ➥ amd64/current/images/netboot/netboot.tar.gz
# tar xf netboot.tar.gz
```

- 可以选择修改 txt.ctf 文件来预设种子参数或自定义超时。见 4.3 节。接下来，可以利用配置管理工具来管理或任意配置远程机器。
- SaltStack 是中心化的配置管理服务：一个 Salt Master 管理多台 Salt minion。将 salt-master 软件包安装在服务器上，salt-minion 软件包安装在被管理主机上。
- 编辑/etc/salt/minion 这个 YAML 格式的配置文件，设置 master key 为 Salt Master 的 DNS 域名或 IP 地址。
- 在/etc/salt/minion_id 文件内，设置 minion 的身份 ID：

```
minion# echo kali-scratch >/etc/salt/minion_id
minion # systemctl enable salt-minion
minion# systemctl start salt-minion
```

- 密钥交换如下所示。在 master 上，手动接受 minion 的身份识别密钥，后续连接自动化执行。

```
master# systemctl enable salt-master
master# systemctl start salt-master
master# salt-key --list all
```

```
Accepted Keys:
Denied Keys:
Unaccepted Keys:
kali-scratch
Rejected Keys:
master# salt-key --accept kali-scratch
The following keys are going to be accepted:
Unaccepted Keys:
kali-scratch
Proceed? [n/Y] y
Key for minion kali-scratch accepted.
```

- 一旦连上 minion，即可从 master 上执行命令。示例如下：

```
master# salt '*' test.ping
kali-scratch:
True
kali-master:
True
master# salt kali-scratch cmd.shell 'uptime; uname -a'
master# salt kali-scratch sys.doc'
master# salt '*' service.enable ssh
[...]
master# salt '*' service.start ssh
[...]
master# salt '*' pkg.refresh_db
[...]
master# salt '*' pkg.upgrade dist_upgrade=True
server# salt '*' cmd.shell 'pkill -f dnmap_client'
```

- 执行模块的完整列表，请参见 `https://docs.saltstack.com/en/latest/ref/modules/all/index.html`。
- 使用 Salt State 文件（可重用的配置模块）来调度操作、收集数据、编排多个 minion 的操作序列、在云架构中管理它们，等等。下面是为了节省时间预定义的一些 Salt 规则：

➡ `https://docs.saltstack.com/en/latest/topics/development/conventions/formulas.html`

- fork 软件包时，首先要确定它是否是你需要处理的任务。它们各有优缺点，审查时需要仔细一些。kali-meta、desktop-base、kali-menu 这些软件包是较常见的选择。具体 fork 一个软件包的过程，比较复杂，也比较难总结。

到目前为止，我们已经讲完了所有基础部分的内容，包括安装、配置、定制及部署 Kali Linux。接下来，会讲述 Kali Linux 在信息安全领域中处于一个什么样的角色。

练习题

练习1——分别配置一个Salt master和minion

如标题所述。

练习2——创建一个Kali代码仓库

请创建一个 Kali 代码仓库，将自己创建的软件包放置其下（SET、kali-menu、内核软件包等）。

练习3——从头创建一个配置软件包

请定制一个软件包，该软件包使用你自己的软件包仓库和 GnuPG 签名密钥，分发 SaltStack 配置文件，推送定制的背景图片，以及向多个 Kali 系统统一提供默认桌面设置。

第11章　安全评估简介

内容

- 安全评估中的 Kali Linux
- 安全评估分类
- 安全评估规范
- 安全攻击类型
- 小结

到目前为止，我们已介绍了 Kali Linux 的许多功能特性，对于 Kali 能做什么、如何使用 Kali 来完成一系列复杂的任务工作，你应该也有了较深入的了解。

在动手操作 Kali 之前，需要大概了解一下安全评估的相关概念。本章会逐步介绍这些概念，并提供一些参考文档，这对你将来使用 Kali 进行安全评估会很有帮助。

首先，在信息系统中，“安全”是指什么？信息系统的安全，通常主要指以下三点。

- **机密性（Confidentiality）：**指未验证状态下的用户，是否可以访问敏感数据或系统。
- **完整性（Integrity）：**指数据或系统是否被非法破坏或修改。
- **可用性（Availability）：**指可被访问的数据和系统，是否可用，是否与其预期一致。

这三者合一，简称为 CIA（Confidentiality、Integrity、Availability），这将是你以后在部署、维护和评估一个安全的系统时，需要关注的主要内容。

需要注意的是，在不同的情况下，需要特别关注 CIA 三元组中的某一个。比如，你有写日记的习惯，里面有许多个人隐私，那么日记的机密性要排在首位，它远比完整性和可用性重要。换句话说，你并不关心是否有人往日记里写东西（而不阅读日记），或者日记是否触手可及随时可读。另一方面，如果需要防护一个医药处方记录系统，数据的完整性则是第一位的。虽然阻止别人查阅其他人使用药物的情况，也很重要，但如果有人更改系统内容（破坏完整性），那么它可能会产生危及生命的后果。

当你在做系统防护时，发现了一个问题，那你首先应考虑，它属于 CIA 三元组中的哪一个或者哪两个组合的问题。这样可以帮助你更全面地理解问题，更准确地归类问题，更迅速

地响应问题。下面使用 Web 应用中的 SQL 注入漏洞，作为例子讲解。

- **机密性：**利用该 SQL 注入漏洞，攻击者可提取 Web 应用程序的数据库内容，对所有数据具有访问权限，但无法更改数据库，或者阻止他人访问数据库。
- **完整性：**利用该 SQL 注入漏洞，攻击者可修改现有数据库的已有信息，无法读取数据库，以及阻止他人访问数据。
- **可用性：**利用该 SQL 注入漏洞，攻击者可启动一个持久运行的查询，消耗大量系统资源。如果多次启动该查询，那么可能导致拒绝服务（DoS）。此时，攻击者不具备数据库的读、写权限，但可阻止合法用户正常访问 Web 应用程序。
- **多者合一：**利用该 SQL 注入漏洞，可导致攻击者获取 Web 应用服务器的交互式 shell。在这种情况下，可任意读取数据，违背机密性；可随意更改数据，破坏完整性；可恣意关停服务，危及可用性。

CIA 的概念并不复杂，在实际工作中，即使没辨别出属于哪种，也可凭直觉进行处理。而重要的是，你需要用心去思考这些概念，用这些概念来指导自己努力的方向。这些基础概念，可以帮你迅速判断什么是系统关键组件，以及在整改现有问题时，哪些工作和资源值得去投入。

另一个需要特别指出的概念是“风险（risk）”，以及风险是如何由威胁（threat）和漏洞（vulnerability）构成的。这些概念并不复杂，但很容易搞错。我们在后面会详细介绍。从宏观上来讲，“风险”就是指你要预防发生的事情；“威胁”指谁要对你做这些事情；“漏洞”指是什么允许他们能够做这些事情。可以通过对“威胁”和“漏洞”的控制，来达到降低“风险”的目的。

举个例子，你在周游列国时，至某些国家地区，你可能要面临感染疟疾的风险。这是因为这些地区蚊子带来的威胁很大，而且人很难对疟疾免疫。但是，你可以通过使用抗疟疾的药物，控制人体抗体这个“漏洞”，使用驱蚊剂和蚊账，控制来自蚊子的“威胁”。通过对“威胁”和“漏洞”的控制，来确保“风险”无法发生。

11.1 安全评估中的Kali Linux

在实战中使用 Kali Linux 时，首先要确保自己的 Kali 是个干净的版本。刚从事安全行业的新手，常犯的错误，就是在多个安全评估工作中，一直使用同一套 Kali 环境。而使用多套 Kali Linux 则是基于以下两点考虑：

- 在安全评估过程中，常常需要手动安装、调整甚至更改系统某些方面。这些临时改动，当时可能会帮助解决某个问题，但很难持续跟踪它们，也使得系统更难以维护，

甚至导致往后的配置越来越复杂。

- 每个安全评估案例都不一样。当时留下的备注、代码及其他改动，可能会互相之间乱成一团；甚至更坏的情况就是，会交叉污染客户的数据。

这就是我们为什么强烈推荐使用纯净 Kali 安装版的原因，前面章节介绍的，如何准备预定制的、便于快速自动化安装的 Kali Linux 系统，在这里就用到了。如何准备，具体参见 9.3 节及 4.3 节的内容，当前自动化程度越高，明天你浪费的时间就越少。

当涉及在自己的工作空间内配置使用 Kali 时，虽然每个人的需求都各有不同，但还是有些普适性的建议：首先就是使用加密的安装方法，详见 4.2.2 一节所述。假如你的笔记本电脑不慎失窃，这将是确保数据安全的救命之法。

如在外差旅，有更高的安全性需求，建议使用核密码加密（如第 9 章所述），随后将核密码在办公室留下一份。这样的话，在你回到办公室使用密码恢复数据之前，数据都是安全的。

另一个需要再次确认的是已安装的软件列表。根据你要执行的任务，考虑安装哪些工具。比如，如果你正着手做无线安全测试评估的工作，则可以考虑安装 kali-linux-wireless 软件包，它包含所有 Kali Linux 中的无线安全测试评估相关的工具；如果需要对某个 Web 应用进行渗透测试，建议安装 kali-linux-web 软件包，它是 Web 应用测试工具的大集合。要做好应对测试评估过程中出现无法访问互联网的情况，所以准备工作越充足越好。

基于同样的考虑，需要去检查你的网络设置（见 5.1 节和 7.3 节）。反复确认 DHCP 设置和监听运行服务。这些设置均会对后续任务的成功与否产生极大影响。毕竟你无法去测评你看不到的系统，而且启动过多的监听服务，也会让系统变得有问题，可能你还没开始干活呢，系统就挂了。

如果你的任务还涉及入侵行为的调查取证，那么需要特别注意你的网络设置，而且尽量避免对目标系统做任何改动。安装有 kali-linux-forensic 软件包的定制版 Kali，可从取证模式启动，在这种模式下，不会自动挂载磁盘，也不会使用交换分区。使用这种模式，可以在使用 Kali 取证工具进行分析时，维持系统的完整性。

针对不同工作任务，恰当地准备相应的 Kali 安装版非常必要。一个干净、高效的 Kali 环境，可以让你更优雅地投入工作。

11.2　安全评估的类型

Kali 环境准备完毕后，下一步需要准确定义你参与的测试评估工作属于哪种类型。从最顶层来看，定义了 4 种类型：漏洞评估、合规性评估、传统渗透测试、应用程序评估。有些

案例可能涉及多种测试评估，因此值得深入去了解它们，以及它们与不同 Kali 版本之间的关联。

在这之前，首先要能区分漏洞（vulnerability）和漏洞利用（exploit）的区别。

漏洞（vulnerability），是指可以用来破坏信息系统机密性、完整性或可用性（CIA）的某个缺陷。漏洞类型有很多，包括但不限于以下几种。

- **文件包含：**文件包含漏洞 [1]，常见于Web应用之中，允许攻击者将本地或远程文件，包含到程序中执行。比如，某个Web应用内，有个“Message of the day”函数，函数功能是读取一个文件的内容，并将其包含到Web网页中，显示给用户。一旦这种功能没有被正确编码实现，那么攻击者即可利用它，修改Web请求，强制网站包含攻击者指定的文件内容。
- **SQL注入：**SQL注入攻击 [2]，是指程序输入验证的地方被绕过，从而使得目标程序，可以执行攻击者提供的SQL语句。这也是注入攻击的一种形式，会导致一系列潜在的安全问题。
- **缓冲区溢出：**缓冲区溢出漏洞 [3]，是指因程序设计缺陷，导致向程序某个缓冲区写入的数据，写进邻近的内存之中。在某些情况下，邻近内存可能对目标程序的执行特别关键，可以通过精巧地构造所覆盖的内存数据，从而达到控制程序执行的目的。
- **条件竞争：**条件竞争漏洞 [4]，是一种利用程序时间间隔的漏洞。在某些情况下，程序的工作流程，依赖于特定的事件顺序。如果事件顺序被更改，则可能会产生一个漏洞。

漏洞利用（exploit），是指利用特定漏洞的一个程序代码，但不是所有漏洞都是可利用的。因为漏洞利用，必然会改变程序运行，产生非预期操作，所以利用程序会相对比较复杂。此外，在现代计算机系统中，也有许多缓解机制，这些机制使得漏洞利用更加困难，比如数据执行保护 [5]（DEP）和地址空间随机化 [6]（ASLR）。然而，不能因为某个漏洞没有公开已知的漏洞利用代码，就判断它不可利用。比如，许多组织销售商业的漏洞利用程序，这些漏洞利用必然不会公开，因此所有的漏洞，均要认为其是潜在可利用的。

1 https://en.wikipedia.org/wiki/File_inclusion_vulnerability
2 https://en.wikipedia.org/wiki/SQL_injection
3 https://en.wikipedia.org/wiki/Buffer_overflow
4 https://en.wikipedia.org/wiki/Race_condition
5 https://en.wikipedia.org/wiki/Executable_space_protection#Windows
6 https://en.wikipedia.org/wiki/Address_space_layout_randomization

11.2.1　漏洞评估

漏洞通常是指通过某种方式可破坏信息系统的机密性、完整性或可用性的一个缺陷。在漏洞评估过程中，你的任务是给出一张“目标环境”的漏洞清单。“目标环境”的概念非常重要，要确保在客户的网络环境和规定的目标范围之内。对“目标环境”之外的漏洞审计，将导致服务中断、客户信任危机甚至司法诉讼等情况。

因为漏洞测试并不复杂，所以通常会作为工作职责，定期常态化地在成熟的环境上执行。大多数情况下，可使用Kali下的自动化工具，比如Vulnerability Analysis[1]或Web Applications[2]，来发现目标环境的在线系统，识别监听服务，枚举服务器系统及软件版本等信息。

这些信息会被用于与已知漏洞的特征做对比。这些特征由数据指纹组合构成，用于表示一些已知漏洞。可使用多个数据指纹，数据指纹越多，识别就越精确。现有数据指纹非常多，包括但不限于以下几种。

- **操作系统版本：**软件在某个操作系统版本上存在漏洞，但在另一个版本上可能没有。基于此，扫描器需要精确确定，目标应用程序所在操作系统具体是哪个版本。
- **补丁级别：**很多时候，操作系统补丁并不会影响版本信息，但依然改变了某些漏洞的响应方式，甚至彻底消除了该漏洞。
- **处理器架构：**很多应用程序，设计支持多种处理器架构，比如 Intel x84、x64，ARM 的各种版本，UltraSPARC 等。在某些情况下，一个漏洞仅存在于特定的架构上，所以这个信息对特征的准确性也很重要。
- **软件版本：**目标软件版本，是识别漏洞的基础信息之一。

以上这些，包括许多其他数据指纹，一起构成漏洞扫描的特征信息。数据指纹匹配得越多，特征就会越精确。当处理特征匹配时，可能还会有以下问题。

- True Positive（真正，TP），特征匹配到了真正的漏洞。这些漏洞结果，是需要被进一步整改修补的，否则，恶意攻击者可能利用这些漏洞，攻击你所在的组织机构或你的客户。
- False Positive（假正，FP），也称误报。特征匹配到的不是真正的漏洞。在测试评估过程中，这些就是令人沮丧的噪声。未经更充分的验证，永远不要把匹配到的结果认为是误报。

1 http://tools.kali.org/category/vulnerability-analysis
2 http://tools.kali.org/category/web-applications

- True Negative（真负，TN），特征没有匹配到任何漏洞，确实也不存在任何漏洞。这是理想中的场景，表示目标环境并不存在任何漏洞。
- False Negative（假负，FN），也称漏报。特征没有匹配到任何漏洞，但事实上存在漏洞。与误报类似，漏报这种情况更糟。在这种情况下，问题存在，但扫描器没有探测到它，你对这个漏洞的存在一无所知。

可以想象，特征的准确性对于结果来说是至关重要的。提供的数据越多，得到的扫描结果就越准确，这也是经身份认证的扫描受欢迎的原因。

一次经身份认证的扫描过程，会向其扫描器提供身份认证的凭证。这使得扫描器对目标比其他情况有更深入的视野。比如，在正常扫描中，你可能只从监听的服务以及服务所提供的功能中，检测到系统的相关信息。而经过系统身份认证，你可以全面审查所安装的所有软件、应用程序补丁、运行中的进程等，这种情况下得到的信息量，是普通扫描无法比的。这种规模的数据，对发现其他手段检测不到的漏洞非常有效。

精心筹划的漏洞评估工作，可反映某个组织中的潜在问题，提供其随时间变化的度量指标。虽然这只是个轻量级的评估工作，但众多组织机构，仍会利用空闲时间定期、自动化进行漏洞扫描，以确保服务正常，不出问题。

如前文所述，为获取更准确的结果，一次漏洞扫描，会检查许多不同的数据指纹。这些检查会对目标系统运行造成负载，并消耗一定带宽。然而，很难知道具体消耗了多少资源，因为这取决于目标系统开放的服务数量和检查的形式。这就是扫描的成本。所以在运行扫描工具之前，要对系统可能消耗多少资源、带来多少负载，有个大致的概念。

扫描线程　很多漏洞扫描程序，都有个“扫描线程数”的选项，这表示同时有多少个并发检查。增大这个数字，将会直接影响测试平台、网络带宽、目标环境的负载。当你为了尽快完成扫描任务，尝试增加线程数时，务必记住这种行为会带来大量负载。

漏洞扫描完成后所发现的问题，通常会与行业标准ID关联起来，比如CVE号[1]、EDB-ID[2]和供应商的报告。这些信息与漏洞CVSS分值[3]一起，用于确定风险级别。除漏报和误报之外，这些风险级别，也是分析扫描结果时要考虑的问题。

因为自动化工具都使用特征数据库来探测漏洞，因此任何对已知特征的轻微偏离，都可能改变结果，同样也可以影响探测到的漏洞的正确性。一个误报，会错误标记一个不存在的漏洞；然而一个漏报，指无法有效地感知、通报某个漏洞。正因为如此，一款扫描器的优

1 ttps://cve.mitre.org
2 ttps://www.exploit-db.com/about/
3 ttps://www.first.org/cvss

劣，取决于其特征规则库。出于这个原因，许多厂商提供多个特征库：一种是面向个人用户的免费版；另一种是面向企业用户的，特征库更加全面，但收费相当昂贵。

漏洞扫描常遇到的另一个问题，就是建议风险等级的有效性。这些风险等级，是在通用基础上评定的，考虑了各种不同因素，比如权限级别、软件系统以及预认证或后认证等。根据自己具体的环境，这些评级可能不靠谱，此时就不应该盲目接受。只有那些熟悉系统和漏洞的人，方可准确地评估风险等级。

风险等级，没有通用的定义约定，一般推荐NIST Special publication 800-30[1]作为风险评级基线标准。NIST SP 800-30 将所发现漏洞的真正风险，以发生可能性和潜在影响两者组合的方式进行定义。

发生可能性（Likelihood of Occurrence）

根据美国国家标准技术研究院（NIST）的说法，发生可能性是基于某种特定的威胁，能利用某个特定漏洞的可能性，可分为低、中、高三级。

- **高：**潜在对手技术水平高、动机强烈，且针对该漏洞已采用的保护措施不够充分。
- **中：**潜在对手技术水平较高、动机强烈，但针对该漏洞已采用的保护措施可能会成功防御攻击行为。
- **低：**潜在对手技术水平较低、缺少动机，且针对该漏洞已采用的保护措施是部分甚至完全有效的。

影响（Impact）

影响的级别，是以在漏洞被利用之后所带来的危害程度来评级的。

- **高：**利用该漏洞，可能造成严重财产损失，严重危及组织机构利益或声誉，甚至造成严重人身伤害，包括生命损失。
- **中：**利用该漏洞，可能造成财产损失，危及组织机构利益或声誉，或造成人员伤亡。
- **低：**利用该漏洞，可能造成一定程度的财产损失，影响组织机构利益或声誉。

总体风险（Overall Risk）

一旦确定发生的可能性和影响之后，即可确定总体风险等级。总体风险同样可分为低、中、高三级，这也为负责系统安全运维的人员提供了指导。

- **高：**强烈要求采用额外措施对该漏洞进行保护。在某些情况下，系统可以继续运行，

1 ttp://csrc.nist.gov/publications/PubsSPs.html#800-30

但必须尽快拿出解决方案。

- **中**：有必要采用额外措施对该漏洞进行保护，应及时拿出解决方案。
- **低**：由系统所有者决定，是否采用额外措施，对该漏洞进行保护，或者他们可以接受风险，不对系统做任何改动。

小结

一个漏洞的风险，由如此多的因素共同组成，所以，由工具直接输出的风险等级，仅可作为组织机构整体风险评估的出发点。

由专业人士分析出具的漏洞评估报告，可以作为其他如合规性渗透测试等测试评估工作的基础。因此，了解如何从这个最初评估中获得最好的结果很重要。

Kali是一款无须额外配置，即可用于漏洞评估工作的优秀平台。在Kali应用程序菜单中，如“信息收集（Information Gathering）”、“漏洞分析（Vulnerability Analysis）”和“Web应用分析（Web Application Analysis）”几大类中，均可以发现大量漏洞评估工具。一些网站，包括上文提到的Kali Linux Tools Listing[1]、Kali Linux官方文档[2]、免费的Metasploit Unleashed[3]课程，均是使用Kali Linux进行漏洞评估的优秀学习资源。

11.2.2 合规性渗透测试

按复杂程度排序，下一个就是合规性渗透测试。这是最为常见的渗透测试形式，因为根据合规性规范，要求政府和行业强制执行。

可能有各种行业的合规性规范，最为常见的应该就是支付卡（第三方支付）行业数据安全标准[4]（PCI DSS），这是由支付卡公司规定的、零售商处理卡支付业务必须遵守的框架。还有一些其他标准，如美国国防信息系统局安全技术实施指南[5]（DISA STIG）、联邦风险和授权管理计划[6]（FedRAMP）、联邦信息安全管理法案[7]（FISMA）等。很多时候，基于多种因素，企业客户要求进行测试评估，或查看最近的评估结果。无论是自行组织，还是授权他人，这些类型的评估工作，统称为合规性渗透测试，简称为“合规性评估”或“合规性检

1 http://tools.kali.org/tools-listing
2 http://docs.kali.org
3 https://www.offensive-security.com/metasploit-unleashed/
4 https://www.pcisecuritystandards.org/documents/Penetration_Testing_Guidance_March_2015.pdf
5 http://iase.disa.mil/stigs/Pages/index.aspx
6 https://www.fedramp.gov/about-us/about/
7 http://csrc.nist.gov/groups/SMA/fisma/

查”。

一次合规性评估通常从漏洞评估开始。以支付卡行业（PCI）合规性审计 [1]为例，如果漏洞评估执行较好，即可满足一些基础需求，包括：2. 不要使用供应商提供的默认系统密码和其他默认的安全参数（如菜单中，密码攻击类下的工具）；11. 定期测试安全系统和进程（如菜单中，数据库评估下的工具）等。有些需求，如：9. 限制对持卡人数据的物理访问、12. 制定面向所有人员的信息安全规章制度，看起来不需要传统的基于工具的漏洞评估。

尽管在合规性检查的某些方面，看起来无法直接使用 Kali Linux 完成，但实际上，Kali 是完美适配这种场景的，不仅因为它集成了大量的安全工具，还因为它基于的开源 Debian 环境，使得各种各样的工具均可安装支持。从合规性框架选取关键字，在包管理器中搜索，你几乎肯定会得到多个结果。当前，许多组织也将 Kali Linux 作为各自行业合规性检查的标准平台。

11.2.3　传统渗透测试

传统渗透测试，取决于其执行环境不同，很难直接定义。之所以混乱，部分是因为“渗透测试（Penetration Test）”一词更常用于上一节提到的合规性渗透测试（甚至是漏洞评估），按其设计，无须太深入评估，否则会超过其最低要求。

在本节，我们使用“传统渗透测试”这一类别，主要进行超出最低要求的评估工作，评估的目的是提升组织机构的整体安全水平。

渗透测试，与之前的评估类型不同，其不是从一个定义的范围开始，而是从一个任务目标开始，比如，“如果一个内部用户出现问题，会发生什么”，或者“如果组织受到来自外部的、集中的恶意攻击行为，会发生什么”。这类评估的关键区别在于，它不仅要发现和验证漏洞，还要揭示利用该漏洞能做到的最坏情况是什么样的。此时，不能仅依赖于漏洞扫描工具集，还要充分利用验证，排除误报，尽力全面检测，找出漏报。这些工作通常包括：利用所发现的漏洞，提升访问权限，利用提升的访问权限，再对系统做额外的攻击。

这需要有较强的动手能力来搜索探测，有创造性的思维能力来发现漏洞，使用扫描器之外的工具来验证测试，从而对目标环境进行严格审查。完成一次之后，通常仍需要按照这个流程，反复执行多次，以确保评估工作细致全面。

按照这种方法，你经常会发现很多评估工作由多个不同的阶段组成。Kali 在菜单中按照这些阶段分类，以便于轻松找到所需工具。

1 https://www.pcisecuritystandards.org/documents/PCIDSS_QRGv3_2.pdf

- **信息收集：**在本阶段，对目标环境了解越多越好。通常，这个阶段的动作是非侵入式的，表现类似于正常的用户活动。这些都为后续的评估工作铺路，因此需要越全面越好。Kali 的“信息收集（Information Gathering）”类别中有几十个工具，可全面满足收集目标环境信息的要求。
- **漏洞发现：**也称为“主动信息收集”，没有攻击行为，但使用非正常用户行为，尝试识别目标环境的潜在漏洞。这也是之前讨论的漏洞扫描器的活。漏洞分析相关的工具，可在“Web 应用分析（Web Application Analysis）”、“数据库评估软件（Database Assessment）”和“逆向工程（Reverse Engineering）”分类中找到。
- **漏洞利用：**本阶段，在发现已知漏洞的基础上，去利用漏洞，从而进入目标系统，拥有立足之地。这一阶段的辅助工具，可在“Web 应用分析（Web Application Analysis）”、“数据库评估软件（Database Assessment）”、“密码攻击（Password Attacks）”和“漏洞利用工具集（Exploitation Tools）”分类中找到。
- **跳板和数据窃取：**获得立足点之后，还有一系列工作需要去做。通常是提权至能够完成任务的级别；通过跳板进入其他无访问权限的系统；从目标系统提取敏感信息。该阶段工具可在“密码攻击（Password Attacks）”、“漏洞利用工具集（Exploitation Tools）”、“嗅探 / 欺骗（Sniffing & Spoofing）”和“权限维持（Post Exploitation）”分类中找到。
- **报告：**评估工作完成之后，需要将评估过程中的动作形成报告文档。与前面相比，本阶段工作技术性不高，重要的是确保客户通过文档从评估工作中，能够获得充分的价值。可在“报告工具集（Reporting Tools）”分类中找到该阶段相关工具。

多数情况下，因组织机构不同、面临的威胁源不同、受保护目标也不同，所以评估工作也不尽相同。Kali Linux 是一个面向不同类型评估工作的全能型选手，这也是可以真正利用 Kali Linux 定制化功能的时候。许多组织机构，针对多种评估类型，会内部分别维护一个高度定制的 Kali Linux 版本，以便于执行新的评估工作时，快速化部署。

组织机构可能对 Kali Linux 进行定制的内容，包括以下这些。

- **预安装商业软件：**例如，你可能有一款商业化的漏洞扫描器，为了避免重复安装，可以执行一次动作 [1]，每个部署版本均可使用。
- **预配置反向连接VPN：**在执行“远程内网”评估工作时，这个功能会非常有用。多数情况下，这些系统会连回评估人员控制的系统来创建隧道，以便于评估人员可直接访

1 http://docs.kali.org/kali-dojo/02-mastering-live-build

问内网系统。Kali Linux ISO of Doom[1]就是这种类型的一个定制版。

- **预安装内部开发工具套件：**许多组织机构均有一些私有的工具套件，在Kali里一次性定制好[2]，可以节省不少时间。
- **预定义OS配置：**如主机映射、桌面壁纸、代理设置等。许多Kali用户有自己偏好的特定设置[3]。如果你打算在正常版本的基础上，重新部署Kali，考虑加上这些配置，会比较有意义。

11.2.4　应用程序评估

大多数评估任务的目标是一个范围，但应用程序评估是针对特定应用程序软件的。因各类组织机构内部核心业务系统的复杂性，这类评估也越来越常见。应用程序评估通常打包在更大的评估任务中。其评估形式，包括但不限于以下几种。

- **Web 应用：**最为常见的外部攻击面，Web 应用因其可从外部访问，所以是个好目标。通常，标准的评估工作，只会发现一些基础的问题，如要得到更多结果，需要花时间去关注应用程序工作流程相关的问题。kali-linux-web 基础包内，有很多在这里使用的工具。
- **桌面编译软件：**不单是服务器软件，桌面应用程序也是非常好的攻击面。在过去，许多桌面应用，诸如 PDF 阅读器、基于 Web 的视频程序都是重点目标，这也倒逼了这些应用更加成熟稳定。然而，依然有相当多的桌面应用程序存在大量漏洞。
- **移动 APP：**随着移动设备的普及，移动 APP 也逐渐呈现更多的攻击面。这是一个快速迭代的目标，该领域技术方法也日新月异，导致几乎每周都有新的开发进展。移动 APP 相关的分析工具，可以在“逆向工程（Reverse Engineering）”下找到一部分。

应用程序评估工作的执行可以有多种形式。一个简单的例子，可针对应用程序，运行特定的自动化工具，以识别潜在问题。这些工具，依赖具体的应用程序逻辑，而不是已知特征码。因此，这些工具，需要对应用程序行为，有深入理解。常见的例子是Web应用扫描器，比如Burp Suite[4]，它首先识别Web应用的各个输入字段，然后向这些字段发送通用的SQL注入攻击语句，同时监控程序响应数据，以判断攻击是否成功。

在更复杂的场景中，应用程序审计可以以黑盒（black box）或者白盒（white box）的形

1 https://www.offensive-security.com/kali-linux/kali-rolling-iso-of-doom/
2 http://docs.kali.org/development/live-build-a-custom-kali-iso
3 https://www.offensive-security.com/kali-linux/kali-linux-recipes/
4 https://portswigger.net/burp/

式，进行交叉式执行。

- **黑盒测试：**工具（或测试人员），不需要对系统特别了解，或在没有标准用户访问权限的情况下，与应用程序进行交互。比如，在一个 Web 应用程序的案例中，测试人员仅可在未登录系统情况下访问那些功能和特性。使用的用户账号均是自行注册的账号。如果用户账号需要由管理员统一创建，这样便能阻止攻击者检测需要用户登录的功能模块。
- **白盒测试：**工具（或测试人员），能够直接获取源码、登录应用承载平台的管理员权限等。可对应用程序的所有功能进行全面且综合的检测，而不论某个功能在应用程序的什么位置。对于白盒评估需要注意的是，它绝不是对恶意行为的真实模拟。

介于以上两种形式之间的又叫灰盒测试（grey box）。通常，具体采用何种形式，取决于评估目的。如果目的是，判断应用程序受到外部集中攻击时发生了什么，那么黑盒可能是最好的选择。如果目的是，在较短的时间内消除尽可能多的安全隐患，那么白盒可能会更有效。

在其他情况下，可能会采用混合的方法，比如测试人员看不到应用程序源码，但管理员给分配了账号，以便于更深入测试尽可能多的功能。

Kali是各类应用程序评估的理想平台。在默认安装版中，已预置大量应用程序扫描器。面向更高级的评估任务，还有一系列的工具、源码编辑器和脚本环境。可参见Kali Tools[1]网站下的“Web应用评估（Web Application Analysis）[2]”和“逆向工程（Reverse Engineering）[3]”两块内容。

11.3 评估规范

随着评估类型的确定，Kali 环境准备完毕，下面即将开始工作了。此前，要做的最后一步，就是规范评估工作。这点很重要，它包括确定工作期望与目标、授予评估工作权限，否则就是非法行为。这里我们会从一个较高的角度来描述，但这毕竟是一个非常复杂且重要的步骤，建议向你所在组织机构的法务代表寻求帮助。

作为规范化流程的一部分，需要定义的工作规范，包括但不限于以下部分：

- 允许访问的系统是哪些？这很重要，可以确保你的行为不会意外干扰到业务系统运行。
- 评估过程中的攻击时间和攻击入口，有没有限制，是哪些？有些组织机构会限制评估

1 http://tools.kali.org
2 http://tools.kali.org/category/web-applications
3 http://tools.kali.org/category/reverse-engineering

任务的工作时间。

- 发现安全漏洞时，是否允许对其攻击利用？如果不允许，审批流程是什么？有些组织机构会严格控制每次的攻击利用，而有组织机构更想贴近实际。所以，最好在工作开始之前，提前确定好。
- 发现重大问题，如何处理？有时客户希望立即得到通告，否则，一般是在评估工作结束后处理。
- 发生紧急情况时，应联系谁？知道发生某个问题应该联系谁是最重要的。
- 谁应该知道这次的评估活动？有时，组织机构希望将员工的事件应急响应能力和安全检查能力，作为评估工作的一部分。事先明确这一点，就可以知道是否需要在评估过程中采取隐蔽行动。
- 评估结束后的期望结果是什么？如何沟通结果？了解各方对评估任务的期望，定义交付，是确保工作完成，让各方都满意的最好方法。

虽然还不够全面，但可以据此清单，去了解具体要做哪些事情。好的法务代表是无可替代的。这些条目一旦确定，需要获得合适授权来执行评估任务，因为没有权威授权许可的情况下，评估过程中的大部分行为都可能不合法。

万事俱备后，还差最好一步：验证。永远不要无条件相信提供给你的目标范围，一定要反复验证。从多种信息来源，确认目标范围内的系统确实为客户所有，并且也是由客户维护的。随着云服务的流行，某些组织机构可能会忘记，他们并不实际拥有为他们提供服务的系统。你可能还需要云服务提供商的特别许可才行。此外，一定要反复验证确认 IP 地址块。即使客户真的拥有整个 IP 地址段，甚至赞同将其列入评估目标，也要验证。比如，我们经历过的一个例子，某组织机构要求对整个 C 段的网络地址进行安全评估，但实际上，他们只拥有这段地址的一个子集。攻击整个 C 段地址空间，最终会攻击该组织机构在网络上相邻的其他机构。“信息收集（Information Gathering）”菜单下的“OSINT 分析（OSINT Analysis）”子分类，包含很多辅助验证阶段使用的工具。

11.4　攻击类型

一旦工作开始，要具体执行哪些攻击？每种类型的漏洞 [1]，均有相应的漏洞利用技术。本节将介绍最常遇到的那些漏洞。

不论你看到什么类型的漏洞，在Kali里相应的工具和漏洞利用都很容易找到。在图形化界

1 https://www.cvedetails.com/vulnerabilities-by-types.php

面的Kali菜单中，工具按类别区分，以便于快速找到合适的工具。此外，Kali Tools网站[1]提供了Kali工具的完整清单，其按类别组织和标记，以便于浏览。每个条目均包含工具的详细信息和使用样例。

11.4.1 拒绝服务

拒绝服务（Denial of Service）攻击，是指利用漏洞造成服务不可用，常见方式是使得有漏洞的进程崩溃。Kali 菜单下的“压力测试（Stress Testing）”分类，包含很多用于此目的的工具。

当大家听到“拒绝服务攻击”时，首先想到的是，对同个目标同时请求大量资源的资源消耗类攻击。它们可能是分布式拒绝服务攻击，或称 DDoS。这种类型的攻击，不属于专业安全评估的内容。

单个拒绝服务攻击，最有可能是未正确利用漏洞导致的结果。如果漏洞利用的编写者，释放了功能不全的利用代码，或称概念验证代码（PoC）。如果你在现场使用这种代码的话，可能就会造成拒绝服务。即使是正确的利用代码，因其只适用特定环境，在其他环境一样会导致拒绝服务。解决办法是，使用安全的、经过充分测试的利用代码，或自己写一个。但即使采用这种方法，也无法完全保证可以避免拒绝服务，且这严重限制了测试人员，造成过度约束，从而导致评估工作完成得不好。此时关键是如何取舍，现场要避免使用 PoC 代码和未测试的漏洞利用代码，以及最好有法务人员帮你处理额外事故。

通常，拒绝服务攻击不是有意引发的。大多数自动化漏洞工具，都会将拒绝服务漏洞定义为低危漏洞，因为虽然你可以终止服务，但服务无法被利用，做到任意代码执行。但要注意的是，并不是所有漏洞利用代码都是公开的，拒绝服务漏洞极有可能造成更深入、更严重的威胁。某些拒绝服务漏洞，其任意代码执行利用脚本可能已经存在，但只是没有公开而已。总之，要小心拒绝服务漏洞，不管漏洞风险级别如何（通常风险级别很低），都要建议你的客户打上补丁。

11.4.2 内存破坏

因编码不当，进程内存空间的某个位置被意外修改，则会发生内存破坏。内存破坏，通常会导致不可预知的程序行为，但多数情况下，这些 bug 可通过控制进程内存的方式，劫持程序执行流，从而允许执行攻击者定义的代码。

1 http://tools.kali.org/tools-listing

这一类的攻击，通常也称为缓冲区溢出（buffer overflow）。这个术语过于简化了，最为常见的几种内存破坏行为，相互差异其实很大，如要成功利用，均有各自的策略和技巧。

- **栈溢出**：程序向某个缓冲区写入的数据，超出缓冲区大小之后，相邻的内存即会被破坏，通常可导致程序崩溃。
- **堆溢出**：堆内存是在运行时分配的，通常包含运行中程序的数据。通过操控堆块链表，进行数据覆盖，即可造成堆内存破坏。
- **整数溢出**：当程序要创建的数字值，超出其类型分配的存储空间时，则发生整数溢出。
- **格式化字符串**：程序接受用户输入，没有检查就将其格式化，取决于使用的格式化标识符，可以造成内存泄露或内存覆盖。

11.4.3　Web漏洞

因现代 Web 站点不再使用静态页面，而是向用户动态生成，故正常的网站已相当复杂。Web 漏洞利用这种复杂性，来攻击后端页面生成逻辑或站点访问者。

这类的漏洞非常普遍，导致很多组织机构，极少敢向外部提供服务。两种最为常见的Web攻击行为[1]分别是SQL注入和跨站脚本（XSS）。

- **SQL 注入**：这类攻击，利用程序编码失误，未对用户输入正确处理，导致可从数据库提取信息，甚至是完全接管服务器。
- **跨站脚本（XSS）**：同 SQL 注入类似，XSS 攻击同样因用户输入未正确处理，允许攻击者操纵用户在自己的浏览器会话中执行任意代码。

庞大、繁复的 Web 应用程序非常普遍，也是最受恶意攻击者偏爱的攻击面。在“Web 应用评估（Web Application Analysis）”菜单及 kali-linux-web 基础包中，有大量相关工具。

11.4.4　密码攻击

密码攻击，是对服务认证系统的攻击。这些攻击通常分为在线和离线两种，相关工具可以在 Kali 菜单的“密码攻击”分类中找到。在线密码攻击，针对正在运行的系统，尝试使用多个密码。离线密码攻击，即已获得密码的哈希值或加密值，攻击者尝试获取密码明文。对这种离线攻击的防护机制是：破解密码的计算复杂性很高，从而限制了每秒尝试的密码数

1 https://www.owasp.org/index.php/Top_10_2013-Top_10

量。然而，变通方法也是有的，比如使用图形处理器（GPU）来增加每秒尝试的次数。kali-linux-gpu 基础包，包含了许多利用显卡计算能力的工具。

最为常见的密码攻击，是针对各类供应商提供的默认密码的攻击。因为这些是已知的，攻击者通过扫描默认账号来碰运气。其他的攻击方法，如自定义字典攻击，创建一个专门针对目标环境的字典，然后对常见的、默认的和已知的账户密码，按顺序进行在线密码攻击。

在评估过程中，了解这类攻击带来的潜在后果很有必要。首先，反复的尝试登录验证，会带来嘈杂的网络环境；第二，无效登录次数过多，经常会导致账户锁定；最后，如果选用字典太大，攻击效率会极低。

11.4.5 客户端攻击

大多数攻击都是针对服务器的，但随着服务越来越难搞，相对容易攻击的目标就被挑出来了。客户端攻击就是其中之一，攻击者的目标是员工安装在工作站上的各类应用程序。Kali 菜单中的“社会工程学工具集（Social Engineering Tools）”分类下，有大量优秀工具，可帮助防守此类型的攻击。

早在 2000 年时，这类攻击目标多是 Flash、Acrobat Reader 和 Java 等。在这种情况下，攻击者会试图欺骗目标访问一个恶意 Web 页面。这些页面包含触发客户端漏洞的代码，从而导致在目标系统上运行恶意代码。

客户端攻击较难预防，需要对用户做大量安全意识教育，并持续更新应用程序以及通过网络层控制来有效降低安全风险。

11.5 小结

本章，我们简要介绍了 Kali 在信息安全领域的作用。强调了工作中使用干净 Kali 环境的重要性，为了保护客户信息使用加密方法的重要性，保护双方利益寻求法务代表支持的重要性。

CIA 的组成部分（机密性、完整性、可用性），是我们在进行系统部署、维护与评估时，所需考虑安全相关的主要方面。这个基础概念，可帮助我们识别系统关键组件，判断整改问题所需投入的工作量和资源。

我们还讨论了几种类型的漏洞，包括文件包含、SQL 注入、缓冲区溢出和条件竞争漏洞。

特征的准确性，对于获取的漏洞评估结果来说至关重要。提供的数据指纹越多，得到的扫描结果就越准确，这也是经身份认证的扫描之所以受欢迎的原因。

因为自动化工具都使用特征数据库来探测漏洞，所以任何对已知特征的轻微偏离，都会改变结果，同样也可以影响探测漏洞的准确性。

我们还讨论了评估的四个分类：漏洞评估、合规性检查、传统渗透测试和应用程序评估。尽管每种类型的评估工作，都可以利用一组核心工具，但很多工具和技术是交叉重叠的。

漏洞评估与其他评估相比，相对简单一些，并且通常由对目标环境自动化发现的一张问题清单组成。在本章，我们讨论了，漏洞是一个可被用于破坏信息系统的机密性、完整性或可用性的缺陷。因为它是基于特征的，所以这种类型的评估，严重依赖于特征的准确性，因此也可能会产生误报和漏报。可在 Kali 的“漏洞分析”和“漏洞利用工具集”菜单中，找到相关的核心工具。

合规性测试，基于合规性操作规范，要求政府和行业强制执行（如 PCI DSS、DISA STIG 和 FISMA 等）。合规性测试通常基于漏洞评估工作。

传统渗透测试是一种更深入的安全评估，其目的是基于真实世界的安全威胁，提升组织机构整体的安全态势。该类型测试，分为几个步骤（与 Kali 菜单结构类似），以成功利用漏洞，且跳至其他机器或网段为最终目标。

应用程序评估（通常是白盒或黑盒），针对的是某个应用程序。所使用的相应工具，可在“Web 应用评估”、“数据库评估软件”、“逆向工程”或“漏洞利用工具集”菜单中找到。

最后还讨论了四种类型的攻击行为，包括：拒绝服务，破坏应用程序的正常行为，使其无法访问；内存破坏，导致进程内存可被操控，从而攻击者最终可执行任意代码；Web 攻击，多使用 SQL 注入或 XSS 之类的攻击技术，对 Web 服务进行攻击；密码攻击，利用密码字典，对服务的用户认证进行攻击。

练习题

练习1——信息安全评估

1. 请解释漏洞与漏洞利用之间的区别与关联。
2. 请解释误报与漏报之间的区别。哪一个更加危险？为什么？
3. 什么是 SQL 注入？
4. 什么是缓冲区溢出？
5. 什么是条件竞争？
6. 什么是文件包含漏洞？

第12章　结尾：未来的路

内容

- 拥抱变化
- 炫一下你刚 Get 的新知识
- 进一步探索

恭喜，你当前对 Kali Linux 系统已相当熟悉，放手去做任何你想做的实验吧。你已发现了它大多数的功能特性，也应知晓它的局限性，以及解决这些限制的方法。

如果你还没有去实践所有功能特性，那就把本书放在手边，以备随时参考之需，且在尝试新功能时，不断加深记忆。记住，多实践、多坚持，技术能力方可提升。Try Harder[1]，像Offensive Security的培训师一样反复练习。

12.1　拥抱变化

像 kali-rolling 这种不断更新的版本，本书的某些部分，必然会有过时的一天。我们尽力保持更新（至少网络版更新），但多数情况下，我们会尝试提供一些通用的、较长时间内仍然有用的解释说明。

换句话说，你要随时做好接受变化的准备，找到任何可能出现问题的解决方案。随着对 Kali Linux 以及其和 Debian 系统之间关系的深入理解，你在遇到困难时，可以向 Kali 和 Debian 社区以及从大量资源（bug tracker、论坛、邮件列表等）那里寻求帮助。

不要害怕提交 bug（见 6.3 节）。你提交了一个好的 bug 后，你可能接着会解决这个问题，或至少找到一个好的解决方案。同时，这也帮助了其他受影响的人。

1 https://www.offensive-security.com/offsec/say-try-harder/

12.2　炫一下你刚Get的新知识

你为自己的 Kali Linux 技能感到骄傲吗？你确信记得真正重要的知识点吗？如果这两个问题，任何一个的回答是“是”，那你应该考虑下，申请 Kali Linux 专业认证（KLCP）。

这是一个综合认证，它确保你明白如何在各种实际案例中，正确地部署和使用 Kali Linux。它会使你简历增色不少，也证明你做好了更进一步的准备。

12.3　进一步探索

本书教会了你许多 Kali Linux 用户应该知道的知识，但为保持精简，还有很多主题没有覆盖到。

12.3.1　系统管理方向

如果你想了解更多系统管理的相关知识，建议你查看 Debian 管理员手册：

➨ `https://debian-handbook.info/get/`

你会发现，我们直接跳过了很多常见 UNIX 服务相关的章节。即使在 Kali 一书中相同的章节，也有许多关于包管理系统的附加提示和注解。

Debian 一书显然更深入地讲述了 Debian 社区及其组织形式。虽然这些知识并不重要，但与 Debian 参与者交流时会非常有用，比如提交 bug。

12.3.2　渗透测试方向

你可能已经注意到了，本书没有教你如何进行渗透测试。但你学到的东西仍很重要，Kali Linux 是最好的渗透测试框架，现在你已做好了充分挖掘 Kali Linux 强大功能的准备了。且你拥有参与 Offensive Security 训练所需的基础 Linux 技能。

如果你觉得还没有做好接受付费课程的准备，你可以从参加Metasploit Unleashed[1]的免费在线培训开始。Metasploit是一款非常流行的渗透测试工具，如果计划学习渗透测试，你应该知道它。

1 https://www.offensive-security.com/metasploit-unleashed/

下一步是遵照Penetration Testing with Kali Linux[1]在线课程学习，这是通往著名的“Offensive Security专业认证（OSCP）”之路。该在线课程，可按自己的节奏学习，但认证考核确实很难，24 小时，在真实环境中，通过独立的VPN网络，手动完成渗透测试。

你做好迎接挑战的准备了吗？

Kali解决方案实现

结课项目1：使用Aircrack-NG

请建立自己的代码仓库，在其中管理 aircrack-ng 的定制更新版。

确保任意 Kali 均可使用该代码仓库和这款 aircrack-ng 更新版。

结课项目2：ISO of Doom

请实现 Kali Linux ISO of Doom，使用一个 Kali 云实例（azure、AWS 等）作为远程服务器。

结课项目3：Kali设备和自动化

请使用树莓派设备运行 Kali 和 hostapd，并作为接入点（Access Point）。

然后构建一个最小化的 Kali U 盘，自生模式启动后运行 responder。

1 https://www.offensive-security.com/information-security-training/